Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matic Jesenovec

# Razvoj izobraževalne spletne aplikacije po metodologiji vitki startup

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: izr. prof. dr. Peter Peer

Ljubljana 2015

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Razvoj izobraževalne spletne aplikacije po metodologiji vitki startup

Tematika naloge:

Najprej predstavite teoretične temelje metodologije vitki startup. Predstavite identificiran problem učenja in pomnenja zaradi informacijske onesnaženosti ter predlagajte rešitev na podlagi ponavljanja v presledkih ter efektivnem izkoristku časa. Razvoja aplikacije se lotite po metodologiji vitki startup. Posamezno iteracijo razvoja ustrezno ovrednotite. V zaključku podajte tudi lekcije, ki ste se jih naučili skozi razvoj.

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matic Jesenovec sem avtor diplomskega dela z naslovom:

*Razvoj izobraževalne spletne aplikacije po metodologiji vitki startup*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Petra Peera,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 4. marca 2015 Podpis avtorja:

*Rad bi se zahvalil svojemu mentorju, izr. prof. dr. Petru Peeru, za podporo, potrpežljivost in dobro voljo ter hkrati za to, da mi je pri delu pustil odprte roke, kar me je pripeljalo do mnogih novih spoznanj. Zahvalil bi se tudi Startup šoli Hekovnik, kjer sem pridobil veliko znanja, ki sem ga vključil v diplomsko delo, ter vsem ostalim, ki so vede ali nevede v kakršni koli meri pripomogli k nastanku diplomskega dela.*

# Kazalo

**Seznam uporabljenih kratic** **xiii**

**Povzetek** **xv**

**Abstract** **xvii**

**1 Uvod** **1**

**2 Metodologija vitki startup** **3**

2.1 Poslovni okvir . . . . . 4

2.2 Razvoj kupcev . . . . . 6

2.2.1 Zanka naredi-meri-spoznaj . . . . . 6

2.2.2 Intervjuji s potencialnimi kupci . . . . . 8

2.2.3 Pristajalna stran . . . . . 9

2.2.4 Najosnovnejši sprejemljivi produkt . . . . . 13

2.2.5 Lijak in ključni kazalniki . . . . . 13

2.2.6 Zasuk . . . . . 16

2.2.7 Ujemanje produkta in trga . . . . . 17

2.3 Agilen razvoj . . . . . 17

**3 Predstavitev problema in rešitve** **21**

3.1 Problem: informacijska onesnaženost . . . . . 21

3.2 Učenje, pomnjenje in ponavljanje v presledkih . . . . . 22

3.3 Upravljanje s časom . . . . . 23

3.4 Rešitev . . . . . 23

**4 Razvojna orodja in tehnologije 25**
4.1 Heroku . . . 25
4.2 Python in Django . . . 27
4.3 HTML5, CSS3, jQuery . . . 28
4.4 Readability API in Pocket API . . . 29

**5 Iteracije poslovnih okvirjev in razvoj kupcev 31**
5.1 Prva iteracija . . . 31
5.1.1 Poslovni okvir . . . 31
5.1.2 Intervjuji s strankami . . . 33
5.2 Druga iteracija . . . 34
5.2.1 Poslovni okvir . . . 34
5.2.2 Intervjuji s strankami . . . 35
5.2.3 Razvoj prototipa . . . 35
5.2.4 Razvoj in lansiranje NSPja . . . 39
5.3 Tretja iteracija . . . 42
5.3.1 Poslovni okvir . . . 42
5.3.2 Razvoj in lansiranje NSPja . . . 42

**6 Razvoj aplikacije 45**
6.1 Načrtovanje . . . 45
6.2 Implementacija . . . 47
6.2.1 Opomniki za ponavljanje v presledkih . . . 50
6.2.2 Dodatek za brskalnik Google Chrome . . . 51
6.3 Testiranje . . . 52
6.4 Vzdrževanje . . . 52

**7 Sklepne ugotovitve 55**
7.1 Učne lekcije . . . 55
7.2 Nadaljnji razvoj . . . 57

**Literatura 59**

# Seznam uporabljenih kratic

| kratica | angleško | slovensko |
|---|---|---|
| **AARRR** | Acquisition, Activation, Retention, Referral, Revenue | Pridobivanje, aktivacija, zadržanje, priporočila, prihodek |
| **ADHD** | Attention Deficit Hyperactivity Disorder | Pomanjkanje pozornosti s hiperaktivnostjo |
| **AJAX** | Asynchronous JavaScript and XML | Asinhroni JavaScript in XML |
| **API** | Application Program Interface | Programski vmesnik |
| **ATARI** | A Trigger, Action, Reward, Investment | Sprožilec, dejanje, nagrada, investicija |
| **CSS** | Cascading Style Sheets | Kaskadne stilske podloge |
| **CTA** | Call To Action | Poziv k akciji |
| **DBaaS** | Database as a Service | Podatkovna baza kot storitev |
| **DOM** | Document Object Model | Objektni model dokumenta |
| **HTML** | HyperText Markup Language | Jezik za označevanje nadbesedila |
| **MMF** | Minimal Marketable Feature | Minimalna tržna lastnost |
| **MTV** | Model-Template-View | Model-predloga-pogled |
| **MVC** | Model-View-Controller | Model-pogled-krmilnik |

| **kratica** | **angleško** | **slovensko** |
|---|---|---|
| **MVP** | Minimum Viable Product | Najosnovnejši sprejemljivi produkt |
| **ORM** | Object-Relational Mapping | Objektno-relacijsko preslikovanje |
| **PaaS** | Platform as a Service | Platforma kot storitev |
| **ROI** | Return On Investment | Donosnost naložbe |
| **SaaS** | Software as a Service | Programska oprema kot storitev |
| **SSL** | Secure Sockets Layer | Sloj varnih vtičnic |
| **URL** | Uniform Resource Locator | Enolični kazalnik vira |
| **XML** | Extensible Markup Language | Razširljiv označevalni jezik |

# Povzetek

Cilj diplomskega dela je razviti najosnovnejši sprejemljivi produkt za koncept inovativne izobraževalne spletne aplikacije in pri tem slediti metodologiji vitki startup, ki pomaga usmeriti razvoj produkta tako, da je ta tržno zanimiv. Z upoštevanjem svetovnih trendov obnašanja potrošnikov, raziskav efektivnega učenja in principov dobre uporabniške izkušnje je bil po večih hitrih iteracijah razvit najosnovnejši sprejemljivi produkt, ki je pokazal nadpovprečno pozitiven odziv trga in možnost za nadaljnji razvoj. Končni izdelek, ki se zaradi uspešnega sledenja principom vitkega startupa precej razlikuje od začetne ideje, uporabnikom omogoča efektivno učenje iz člankov na internetu s pomočjo enostavnega shranjevanja pomembnih delov člankov in ponavljanja teh izpiskov po metodi ponavljanja s presledki, ki dokazano bistveno poveča pomnjenje.

**Ključne besede:** podjetništvo, vitki startup, učenje, ponavljanje s presledki, aplikacija.

# Abstract

**Title:** Development of educational web application based on lean startup methodology

**Abstract:** The aim of the thesis is to develop a minimum viable product for the concept of an innovative educational web application and follow the lean startup methodology, which helps focus the development and makes the result interesting for the market. By following global trends in consumer behavior, effective learning research and principles of good user experience a minimum viable product was developed, which showed an above-average positive response from the market and the potential for further development. The final product, which is due to lean startup principles quite different from the initial idea, helps users to effectively learn from articles on the internet by enabling them simple saving of important parts of articles and re-reading them by following a learning method called spaced repetition, which is proven to significantly improve memorization.

**Keywords:** entrepreneurship, lean startup, learning, spaced repetition, application.

# Poglavje 1

# Uvod

Vedno več aplikacij se seli iz namiznega okolja v spletni brskalnik. Razlogov za to je mnogo. Za razvijalce predstavlja poenostavljen razvoj, saj se lahko osredotočijo zgolj na eno platformo, omogoča hitrejše direktno posodabljanje, enostavnejšo analitiko in spremljanje uporabnikov itd. Končni uporabniki pa se izognejo nadležnim in dolgotrajnim namestitvam na računalnik, do aplikacije lahko dostopajo iz katerekoli naprave z dostopom do interneta, podatke imajo ves čas varno spravljene v oblaku, izognejo se ročnemu posodabljanju in ostalim vzdrževalnim opravilom itd.

Aplikacije, ki domujejo na spletu, imajo torej lahko številne prednosti, zato jim v zadnjih letih popularnost strmo narašča. Posledično se je razvil in postal zelo popularen model programske opreme kot storitve (angl. *Software as a Service – SaaS*), ki razvijalcem omogoča enostaven razvoj in vzdrževanje centralno locirane programske opreme, do katere uporabniki običajno dostopajo preko spletnega brskalnika. Tudi sistem monetizacije se je v tem primeru spremenil. V nasprotju s klasično programsko opremo, kjer gre običajno za enkratno plačilo, se v primeru SaaS plačilo največkrat obračuna po sistemu naročnine v mesečnih, pol-letnih ali letnih intervalih.

Zaradi vedno bolj enostavnega razvoja spletnih aplikacij, lahko dostopnega znanja ter vedno večje količine brezplačnih (ali vsaj izjemno cenovno dostopnih) sredstev, katerih se danes lahko poslužimo za razvoj, delovanje

in vzdrževanje spletnih aplikacij, je tovrstna aktivnost med razvijalci – tako podjetji kot posamezniki – v zadnjih letih postala zelo popularna. Posledično se vse pogosteje dogaja, da razvijalci naletijo na neodziven trg. Trg je namreč nasičen s produkti in v kolikor pred in med razvojem ne izvajamo neprestanega testiranja na trgu, lahko zelo hitro razvijemo produkt, ki v realnem svetu za nikogar ni uporaben.

Cilj tega diplomskega dela je razvoj inovativne spletne aplikacije, namenjene sodobnemu načinu učenja, ki bo ob zaključku razvoja zanimiva za trg. Med razvojem smo sledili metodologiji vitki startup, ki nam omogoča hitro ter cenovno ugodno raziskavo trga in testiranje ideje, še predno vložimo veliko količino časa, denarja in ostalih sredstev v razvoj produkta.

# Poglavje 2

# Metodologija vitki startup

Razvijalci programske opreme, tako posamezniki kot zagonska podjetja, imajo običajno predvsem v začetni fazi izjemno omejena sredstva; od finančnih sredstev, ki jih v začetku brez toka prihodkov ni, do časa, ki ga morajo izkoristiti zelo pametno, da jih konkurenca ne prehiti, ter človeških virov, kateri v začetku običajno predstavljajo le krog prijateljev in znancev.

Verjetnost, da bi inovativen produkt, za katerega moramo razviti popolnoma nov poslovni model, v takšnih ekstremnih razmerah lahko zaživel, je zelo nizka. Po nekaterih raziskavah je neuspešnih preko 90 % startup projektov [1]. Z uporabo določenih metod in tehnik pa se lahko razvoja produkta lotimo bolj taktično in svojo intuicijo podkrepimo s podatki iz trga ter tako drastično zvišamo možnosti za uspeh na trgu.

Metodologija vitki startup (angl. *lean startup*) je bila razvita v Silicijevi dolini. Njen pobudnik je Eric Ries, serijski tehnološki podjetnik, ki je na podlagi svojih podjetniških izkušenj želel svetovati ostalim podjetnikom kako v začetni fazi alocirati svoja sredstva na najbolj učinkovit način. Vitko metodologijo je razvil na podlagi japonskega koncepta vitke proizvodjne (angl. *lean manufacturing*); ta skuša maksimizirati prakse, ki ustvarjajo vrednost, ter eliminirati potratne prakse [2].

Za uspeh na trgu potrebujemo uporabnike. To so posamezniki ali podjetja, katerim s svojim produktom rešujemo nek problem. Vitka metodologija

temelji na eksperimentih, komunikaciji z uporabniki ter neprestanem testiranju in izpopolnjevanju produkta tako, da ta ustreza potrebam uporabnikov.

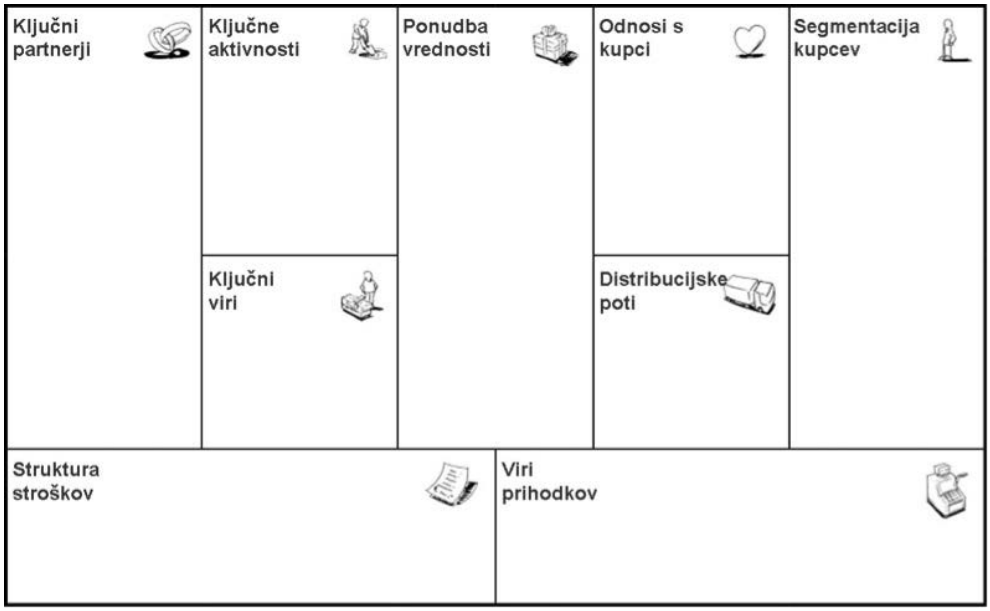
Metodologija vitki startup je bila v osnovi razvita leta 2008 in je bila sprva osredotočena predvsem na visokotehnološka podjetja. Skozi leta se je razvila v filozofijo, kateri danes sledijo posamezniki, ekipe, zagonska podjetja, korporacije, nevladne organizacije in celo nekatere državne ustanove [3].

## 2.1 Poslovni okvir

Prvi korak pri razvoju poslovne ideje je poslovni okvir, ki je v vitki metodologiji nadomestil poslovni načrt. Tradicionalni poslovni načrt je več-deset strani dolg dokument, ki obsega številna dokaj natančna predvidevanja in napovedi o trgu, poslovanju, konkurenci, razvoju, tveganjih, človeških virih, itd.

Po vitki filozofiji poslovni načrt vsebuje mnogo preveč teorije, ugibanj in neakcijskih elementov, zato je Alexander Osterwalder kot alternativo predstavil poslovni okvir (angl. *Business Model Canvas*). Nekatere prednosti poslovnega okvirja pred poslovnim načrtom so:

- Hiter razvoj: namesto večih dni pisanja lahko prvo iteracijo poslovnega modela sestavimo v eni uri.
- Aktivno posodabljanje: po tem, ko v poslovni okvir vnesemo prve hipoteze, jih neprestano skušamo validirati in se tako bližamo k realnemu stanju.
- Fokus na učenju: klasični poslovni načrt skuša dokazati uspešnost posla, poslovni okvir pa – ravno obratno – skuša najti pomanjkljivosti in napake, iz katerih se naučimo, kaj moramo spremeniti.
- Pridobivanje podatkov iz realnega sveta: poslovni okvir spodbuja stalne raziskave, hitre eksperimente in neprekinjen iterativen razvoj produkta, ki omogočajo konstantno izboljševanje.




Slika 2.1: Poslovni okvir.

Poslovni okvir, prav tako kot poslovni načrt, vsebuje vse bistvene parametre, ki jih moramo upoštevati pri razvoju tržno zanimivega produkta. Pri tem se osredotoča na učinkovitost, realne podatke in konstanten razvoj. Do danes je razvitih več različic poslovnega okvirja, najbolj pogosto pa je v uporabi oblika Alexandra Osterwalderja, pri kateri je leva polovica osredotočena na produkt, desna pa na trg (slika 2.1). Okvir vsebuje sledeča polja [4]:

- Segmentacija kupcev: kdo so naši najpomembnejši kupci, katerim ustvarjamo največjo vrednost?
- Odnosi s kupci: kakšen tip odnosa je najbolj primeren za vsakega izmed segmentov kupcev?
- Distribucijske poti: katere distribucijske poti so najbolj primerne, zanesljive in učinkovite za vsakega izmed segmentov kupcev?
- Ponudba vrednosti: kaj je problem, ki ga rešujemo, in rešitev, ki jo ponujamo?

- Ključne aktivnosti: katere ključne aktivnosti zahtevajo naša ponudba vrednosti, naše distribucijske poti, naši odnosi s kupci in naši viri prihodkov?
- Ključni viri: katere ključne vire zahtevajo naša ponudba vrednosti, naše distribucijske poti, naši odnosi s kupci in naši viri prihodkov?
- Ključni partnerji: kdo so naši ključni partnerji in izvajalci ter katere vire nam zagotavljajo in katere aktivnosti prodobivamo od njih?
- Struktura stroškov: kateri so najpomembnejši stroški ter najdražji viri in aktivnosti?
- Viri prihodkov: koliko so kupci pripravljeni plačati ter kako?

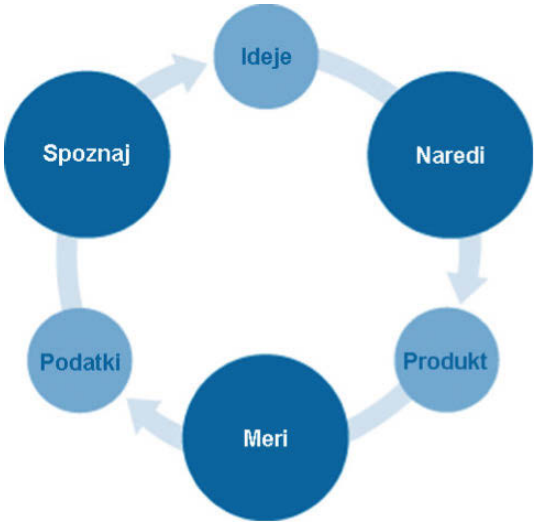
Poslovni okvir je orodje, ki nam pomaga pri razmišljanju, da med razvojem produkta oz. podjetja upoštevamo vse pomembne dejavnike. Priporočeno je, da ga natisnemo na velik list papirja ter nanj s samolepilnimi lističi lepimo posamezne parametre. To nam omogoča, da imamo poslovni okvir, ki je temelj našega posla, vedno pred seboj in ga lahko enostavno posodabljamo.

## 2.2 Razvoj kupcev

Ker je poslovni okvir le skupek hipotez, moramo te hipoteze sčasoma potrditi ali zavreči, da maksimiziramo možnosti za uspeh produkta. Tega se lahko lotimo na več načinov, odvisno od produkta ter faze, v kateri smo. Proces imenujemo razvoj kupcev (angl. *customer development*), pri tem pa si večinoma pomagamo s sistematičnim testiranjem, rezultat katerega je t.i. validirano učenje.

### 2.2.1 Zanka naredi-meri-spoznaj

Neprestano sistematično učenje je eno izmed glavnih načel vitke metodologije. Najbolj učinkovito in efektivno se ga lotimo z delom v zanki naredi-meri-




Slika 2.2: Zanka naredi-meri-spoznaj.

spoznaj (angl. *build-measure-learn loop*), ki preko zaporednega iterativnega razvoja, analiziranja in učenja razvoj ideje požene v konstanten tok. Zanka obenem narekuje hitrost razvoja ideje – hitreje kot se premikamo od faze do faze, hitreje napredujemo z razvojem v pravo zmer, saj s tem procesom potrjujemo veljavne hipoteze in zavračamo neveljavne. Tako najhitreje minimiziramo potratne aktivnosti – to so vse aktivnosti, ki porabljajo sredstva in ne ustvarjajo vrednosti [5].

Zanko naredi-meri-spoznaj sestavljajo 3 faze (slika 2.2):

- Naredi: v tej fazi ideje pretvorimo v dejanja – to lahko pomeni karkoli, od intervjuja s stranko do implementacije nove funkcionalnosti v produktu.

- Meri: odzive izmerimo z različnimi metodami – to so lahko zapiski intervjuja, analitična orodja, ipd.

- Spoznaj: iz podatkov, ki smo jih zbrali v prejšnji fazi, pridemo do spoznanj, na podlagi katerih lahko potrdimo ali ovržemo naše hipoteze in pridemo do novih idej.

### 2.2.2 Intervjuji s potencialnimi kupci

Prvi korak, ki ga lahko naredimo, da preverimo zanimanje za svojo idejo na trgu, je pogovor s potencialnimi kupci. Intervjuje s kupci lahko opravimo brezplačno in z minimalno količino predpriprav. Ob tem moramo biti pozorni, da se od potencialnih kupcev ne naučimo napačnih stvari. Ljudje smo namreč pogosto (tudi nenamerno) lahko zelo pristranski, zato moramo pri intervjujih s strankami vedno paziti, da zastavljamo pravilna vprašanja in si odgovore pravilno interpretiramo.

Nekaj ključnih pravil in nasvetov za efektivne intervjuje s strankami [6]:

- Pomembna vprašanja, ki jih moramo zastaviti, so tista, katerih odgovori lahko popolnoma spremenijo naš produkt ali poslovni model.
- Na začetku pogovora moramo podati čim manj informacij o svojem produktu.
- Ignorirati moramo pohvale, saj so te pogosto izrečene zgolj iz prijaznosti ali trenutnega navdušenja.
- Pogovarjati se moramo o njihovem vsakdanjem življenju in konkretnih problemih, s katerimi se soočajo; ne o idejah, željah in predvidevanjih.
- Ko dobimo generične odgovore, moramo vedno zahtevati specifične primere in podatke.
- Če rašujemo nek problem, s katerim se soočajo, moramo ugotoviti, kako trenutno rešujejo ta problem.
- Če izrazijo neko konkretno željo, moramo poiskati razlog za to željo, ki tiči v ozadju.

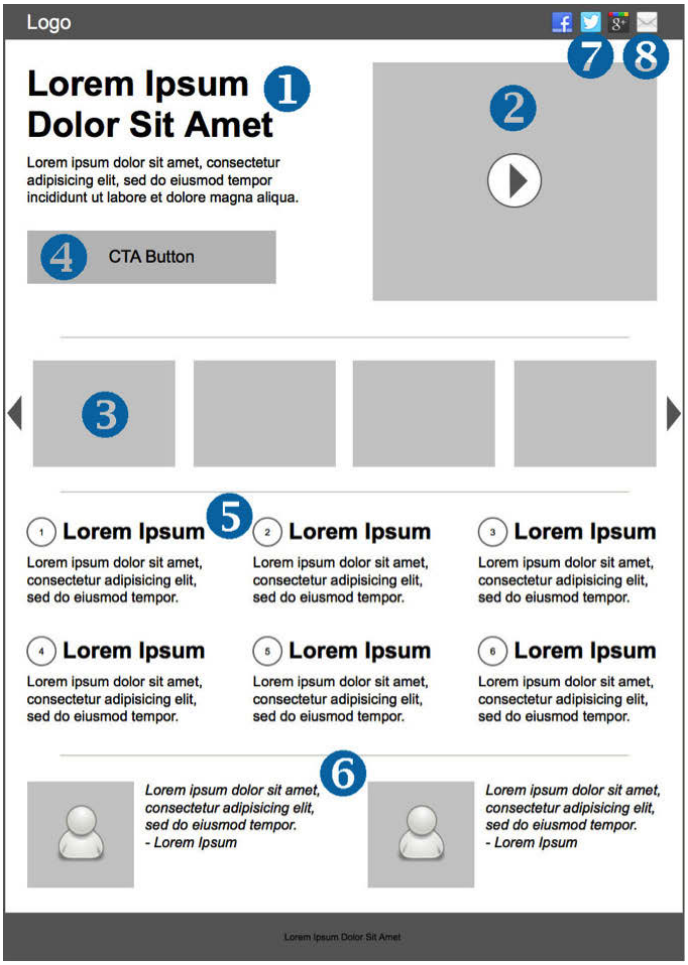
Najbolje je, če intervjuje lahko izvajamo v paru, tako se ena oseba lahko osredotoči zgolj na pogovor, druga pa na zapiske. V kolikor intervjuje izvajamo posamezno, si lahko pomagamo s snemalnikom zvoka, ali še bolje s kamero, saj se včasih lahko veliko naučimo tudi iz govorice telesa sogovornika.

Iz intervjujev s strankami se naučimo, kakšne probleme imajo, kako jih rešujejo, kaj jih pri tem najbolj ovira ter včasih celo, kakšno vrednost jim predstavlja potencialna rešitev. Izredno pomemben del pogovorov s strankami je tudi učenje jezika naših uporabnikov – pogosto namreč strokovnjaki pričnemo uporabljati žargon, ki nam postane samoumeven, medtem ko naši kupci, ki so pogosto laiki na tem področju, uporabljajo popolnoma drugačne izraze. V kasnejši fazi, ko postavimo pristajalno stran (poglavje 2.2.3) ter pričnemo s trženjem produkta, je uporaba kupcem poznanega jezika bistvenega pomena. Zelo znan in nazoren primer uporabnikom prijaznega jezika je uporabil Steve Jobs, ko je 23. oktobra 2001 predstavil iPod. Med predstavitvijo produkta je namesto tehnične specifikacije, da ima naprava trdi disk s 5 GB prostora, raje rekel, da na iPod lahko shranimo 1000 pesmi oziroma celotno glasbeno zbirko marsikaterega posameznika [7].

### 2.2.3 Pristajalna stran

Naslednji korak pri sistematičnem testiranju je običajno pristajalna stran (angl. *landing page*). To je enostavna spletna stran, na kateri pristanejo naši potencialni kupci, mi pa jim skušamo "prodati" produkt, še preden ta zares obstaja. Seveda v tem primeru za produkt običajno ne moremo zaračunati (v kolikor ne zbiramo prednaročil), lahko pa zbiramo kontaktne podatke potencialnih kupcev ali pa obiskovalce prosimo za deljenje novice preko socialnih omrežij. Lahko tudi dodamo nedelujoč gumb z napisom "Kupi" in z analitičnimi orodji štejemo klike na ta gumb. Vse to potrdi zainteresiranost za naš produkt – predvsem merodajna je lahko zadnja tehnika, saj klik na gumb "Kupi" pomeni, da so obiskovalci pripravljeni za naš produkt odšteti denar.

Dobra pristajalna stran mora biti enostavna in razumljiva. Borimo se za vsako sekundo obiskovalčeve pozornosti, saj je razpon pozornosti (angl. *attention span*) uporabnikov na internetu izredno kratek – po raziskavah sodeč je v zadnjih nekaj letih padel na zgolj 8 sekund [8]. Obiskovalec lahko v vsakem trenutku, ko mu nekaj ni všeč ali mu je nerazumljivo, zapusti spletno stran in morda ga za vedno izgubimo. Obiskovalcu mora biti takoj jasno, kaj




Slika 2.3: Model dobre pristajalne strani. [9]

mu skušamo predstaviti, kako mu bo to olajšalo življenje ter kateri je naslednji korak, ki ga mora storiti, da bo to stvar dobil. Za optimalno konverzijo je priporočljivo, da pristajalna stran vsebuje sledeče elemente (slika 2.3):

1. Naslov: Najboljše je, če že naslov izraža ponudbo vrednosti (angl. *value proposition*) za kupca, saj je to običajno prvi element, ki mu obiskovalec strani posveti svojo pozornost.

2. Video: Video posnetek poenostavi predstavitev produkta in omogoča najvišjo vključenost (angl. *engagement*) obiskovalcev, saj istočasno aktivira vizualni in avditivni tip čutenja.

3. Slike: Čeprav aktivirajo le vizualni tip čutenja je prednost slik v tem, da jih vidimo takoj, brez pritiska na gumb za predvajanje. Poleg tega je običajno lažje, hitreje in ceneje ustvariti slike – te so pri digitalnih produktih pogosto lahko le posnetki zaslona.

4. Poziv k akciji (angl. *Call To Action – CTA*): Običajno gre za izstopajoč gumb s tekstom v velelniku, ki od obiskovalca zahteva neko akcijo – npr. nakup ali prijavo na e-novice. To je korak, za katerega si želimo, da ga stori vsak obiskovalec, in je pokazatelj interesa za naš produkt.

5. Značilnosti (angl. *features*): V zgolj nekaj preprostih in kratkih odstavkih opišemo glavne lastnosti in funkcionalnosti produkta. Pri tem uporabljamo jezik naših strank (ki smo se ga naučili preko intervjujev) in upoštevamo kratek razpon pozornosti uporabnikov interneta.

6. Priporočila (angl. *testimonials*): Priporočila služijo kot družbeni dokaz (angl. *social proof*), ki je eden izmed najmočnejših dejavnikov, ko se odločamo o nakupu nekega produkta. Največjo težo imajo priporočila strokovnjakov in slavnih oseb, saj ljudje mnenje nekoga sodimo glede na to, kaj si v splošnem mislimo o tej osebi. A tudi priporočila neznanih uporabnikov imajo lahko zelo močan vpliv, za večjo kredibilnost pa jim lahko dodamo še fotografijo avtorja.

7. Ikone socialnih omrežij: Povezave do socialnih omrežij imajo dve pomembni vlogi. Uporabnike opomnijo in jim poenostavijo deljenje našega produkta preko svojih profilov na socialnih omrežjih. Nam pa tako omogočijo brezplačno promocijo preko priporočila (angl. *referral*), ki imajo izredno močan družbeni vpliv, saj mnenju znancev zaupamo bolj kot mnenju nepoznanih oseb. Poleg tega si uporabniki pred sprejetjem odločitve radi ogledajo, kaj na socialnih omrežjih o našem produktu menijo ostali – tudi v tem primeru govorimo o družbenem vplivu, kjer vseh uporabnikov sicer ne poznamo, zato pa kredibilnost poviša večja količina objav [10]. Konkretna izbira socialnih omrežij je odvisna od ciljne publike.

8. Kontakt: Pristajalna stran je odlična priložnost, da stopimo v direkten kontakt z novimi potencialnimi strankami. Obiskovalci, ki bodo toliko angažirani, da nas bodo sami kontaktirali, si definitivno zelo želijo našega produkta, zato se od njih lahko zelo veliko naučimo.

Tako kot sam produkt tudi pristajalno stran postopoma posodabljamo na podlagi validiranega učenja. Pri tem se poleg odziva uporabnikov v veliki meri zanašamo na analitična orodja (poglavje 2.2.5). V kolikor na strani pristane dovolj obiskovalcev in se zadosten delež le-teh odzove na poziv k akciji, to štejemo kot potrditev zanimanja na trgu. Pri tem lahko dodatno eksperimentiramo z različnimi objavami, ko promoviramo pristajalno stran, da vidimo, na kakšne objave dobimo več odziva.

Namesto klasične pristajalne strani v zadnjih nekaj letih opažamo trend uporabe spletnih platform za skupinsko financiranje (angl. *crowdfunding*), kot so npr. Kickstarter (www.kickstarter.com), Indiegogo (www.indiegogo.com) in Crowdfunder (www.crowdfunder.com). Te nam poleg potrditve ideje omogočajo tudi zbiranje finančnih sredstev in običajno ponujajo enostavno platformo, ki že vključuje vse elemente uspešne pristajalne strani. Predstavljajo pa tudi določene izzive – ker jih uporabljajo predvsem zgodnji uporabniki (angl. *early adopters*), ni nujno, da uspešna kampanja potrdi uspeh

produkta med širšo množico uporabnikov. Da bi kampanja bila uspešna, moramo običajno precej sredstev vložiti v promocijo, da se ne izgubi med ostalimi produkti; običajno moramo za uspešno predstavitev produkta na teh platformah imeti že razvit vsaj najosnovnejši sprejemljivi produkt (poglavje 2.2.4) [11].

### 2.2.4 Najosnovnejši sprejemljivi produkt

Iz pristajalne strani se lahko naučimo, ali za naš produkt obstaja zanimanje na trgu in morda naberemo nekaj elektronskih naslovov potencialnih uporabnikov. Za tem pa je čas za izdelavo produkta. Še vedno nadaljujemo s testiranjem in validiranim učenjem, zato namesto produkta s celotnim naborom funkcionalnosti najprej razvijemo najosnovnejši sprejemljivi produkt oz. NSP (angl. *Minimum Viable Product – MVP*).

NSP vsebuje le glavne lastnosti in funkcionalnosti (angl. *core features*), ki so bistvenega pomena za obstoj produkta. Običajno so to funkcionalnosti, ki v poslovnem okvirju predstavljajo ponudbo vrednosti produkta. Paziti moramo, da se ne zapletemo v idealistično mišljenje in prekomerno razvijemo NSP, saj to lahko predstavlja nepotrebno izgubo časa. Po besedah Reida Hoffmana, ustanovitelja socialnega omrežja LinekdIn: "V kolikor te ni sram prve verzije produkta, si ga lansiral prepozno." [12]

Bistvo NSPja je še vedno učenje, ne optimizacija. Zaradi tega ne optimiziramo programske kode in izkoristka strojne opreme, pomembno je le, da čim hitreje pridemo do delujočega produkta in z njim potrdimo nove hipoteze v poslovnem okvirju. Učimo se lahko na podlagi analitike (poglavje 2.2.5) ali pa nadaljujemo intervjuje s kupci (poglavje 2.2.2), v katere tokrat vključimo praktično uporabo NSPja. [13]

### 2.2.5 Lijak in ključni kazalniki

"Česar ne moremo meriti, ne moremo upravljati." je rek, ki se ga je pametno držati tudi pri razvoju novega produkta. Pri spletnih aplikacijah je merjenje

ključnih kazalnikov (angl. *key metrics*) precej preprosto, saj je za to na voljo veliko orodij, nekaterih celo brezplačnih. Med bolj pogoste sodijo:

- Google Analytics: merjenje osnovnih agregiranih kazalnikov, kot so število obiskovalcev, dolžina obiska, število obiskanih strani, število ponavljajočih obiskovalcev, najbolj obiskane strani ipd.
- Crazy Egg: s pomočjo toplotnih zemljevidov (angl. *heatmap*) in podobnih orodij omogoča analiziranje, kam obiskovalci klikajo, kje premikajo miškin kazalec ter na katerih odsekih strani se največ zadržujejo.
- KISSmetrics: z osredotočenostjo na interakcije individualnih uporabnikov pomaga bolje spoznati obiskovalce kot posameznike; omogoča tudi analiziranje lijaka (angl. *funnel*), A/B testiranje, kohortno analizo ipd.
- Mixpanel: omogoča podobno analizo kot KISSmetrics, a je nekoliko bolj odprt za razvijalce in osredotočen na mobilne naprave.

Če imamo le pristajalno stran, nas običajno najbolj zanima, kakšno je bilo konverzijsko razmerje (angl. *conversion rate*), torej razmerje med vsemi obiskovalci ter tistimi, ki so se odzvali na poziv k akciji. S toplotnimi zemljevidi lahko raziskujemo, kaj obiskovalce na strani najbolj zanima ter kje jih izgubimo. Lahko izvedemo tudi A/B test, kar pomeni, da polovica obiskovalcev vidi A verzijo strani (kontrola), druga polovica pa nekoliko drugačno B verzijo (variacija). Tako lahko na strani testiramo večje ali manjše spremembe (npr. poudarimo drugačno ponudbo vrednosti, testiramo drugačno pozicioniranje pozivov k akciji ipd.) in analiziramo, kako se na njih odzovejo obiskovalci. V kolikor obiskovalce dobivamo iz večih virov (npr. oglasi, forumi, iskalniki), jih lahko segmentiramo glede na vir ter analiziramo, kako se glede na to obnašajo – kateri vir ima višjo konverzijsko razmerje, kateri obiskovalci se dlje zadržujejo na strani ipd.

Ko imamo razvit NSP nas običajno zanima celoten lijak (angl. *funnel*) – ta nazorno pokaže, kateri viri prometa imajo najvišjo donosnost naložbe (angl. *return on investment – ROI*) ter kje izgubljamo uporabnike, iz česar lahko


Slika 2.4: AARRR lijak.

sklepamo, katere korake moramo izboljšati oz. kje so naše hipoteze napačne. Za lijak je pogosto v uporabi ogrodje AARRR, znano tudi pod imenom Kazalniki za pirate (angl. *Metrics for pirates*), ki si ga je zamislil Dave McClure, investitor in soustanovitelj sklada tveganega kapitala 500 Startups. Ogrodje AARRR je sestavljeno iz sledečih petih stopenj [14] (slika 2.4):

- Pridobivanje (angl. *Acquisition*): uporabniki, ki obiščejo našo stran iz različnih virov.
- Aktivacija (angl. *Activation*): uporabniki, ki se registrirajo oz. uporabijo eno izmed bistvenih funkcionalnosti aplikacije.
- Zadržanje (angl. *Retention*): uporabniki, ki se vračajo.
- Priporočila (angl. *Referral*): uporabniki, katerim je produkt tako všeč, da ga priporočijo prijateljem.
- Prihodek (angl. *Revenue*): uporabniki, ki so plačali za produkt.

Število uporabnikov se po lijaku navzdol seveda niža. Končne številke so lahko mnogo nižje, kot sprva pričakujemo. Po izkušnjah Googlovega strokovnjaka za analitiko, Avinasha Kaushika, je povprečno razmerje konverzije v

ZDA 2 %, ne glede na to, če prodajamo slone ali iPode [15]. To velja za razmerje med vsemi obiskovalci, ki obiščejo našo stran, ter tistimi, ki postanejo plačljivi uporabniki.

Priporočila so zelo zanimiv način pridobivanja novih kupcev, na katerega marsikdo pozabi. Kadar za produkt nimamo veliko finančnih sredstev, lahko s priporočili brezplačno generiramo velike količine kvalitetnega prometa. Kvaliteta priporočenih obiskovalcev je visoka iz psiholoških razlogov – ker ljudje priporočilom prijateljev ali pomembnih oseb zaupamo mnogo bolj kot oglasom ali objavam neznanih oseb, je mnogo bolj verjetno, da se bodo takšni obiskovalci konvertirali v uporabnike.

Priporočila lahko spodbudimo na več načinov. Obstoječe uporabnike moramo na vsakem smiselnem koraku opozarjati, da naš produkt delijo preko socialnih omrežij ali elektronske pošte. Deljenje lahko dodatno spodbudimo z nagradami za uporabnike (npr. za vsakega novega uporabnika, ki se registrira preko njih, dobijo 1 mesec brezplačne uporabe produkta), nove registracije preko priporočil pa spodbudimo z nagradami za obiskovalce (npr. tudi vsak novi uporabnik dobi 1 mesec brezplačne uporabe, če se registrira preko priporočila). Predvsem pa velik faktor pri priporočilih igra kvaliteta in inovativnost produkta – izjemno dober produkt v kombinaciji s pravimi uporabniki bo brez težav dosegel viralno rast preko priporočil. [16]

### 2.2.6 Zasuk

Ko pridemo skozi zanko naredi-meri-spoznaj z ugotovitvijo, da je ena izmed naših glavnih hipotez (angl. *core hypothesis*) neveljavna, se na podlagi preostalega validiranega učenja odločimo za zasuk (angl. *pivot*). Zasuk običajno zahteva izdelavo novega NSPja in postavitev nekaterih novih hipotez, ki jih moramo na novo testirati.

Pri zasuku pogosto celoten produkt osredotočimo na le eno funkcionalnost prvotne ideje ali pa obratno, celotno prvotno idejo spremenimo v le eno funkcionalnost bolj obsežnega produkta. Lahko se tudi osredotočimo na drugačen segment kupcev, na drugačen distribucijski kanal, ali pa se odločimo

za drugačen tehnološki pristop k problemu.

### 2.2.7 Ujemanje produkta in trga

Ko potrdimo vse bistvene hipoteze v poslovnem okviru, rečemo, da smo dosegli ujemanje produkta in trga (angl. *product-market fit*). To pomeni, da nam je znano, kdo so naše stranke, kakšen je njihov problem ter kako jim mi ta problem rešujemo. Za profitabilen posel pa nam mora biti poznan tudi skalabilen in zanesljiv model prodaje produkta. [17]

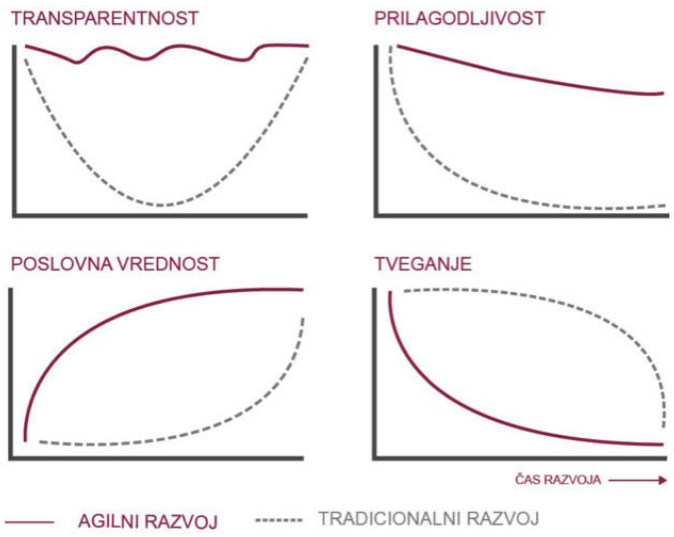
Ena izmed bolj popularnih in enostavnih definicij za merjenje ujemanja produkta in trga je t.i. Pravilo 40 %. Ujemanje dosežemo, ko 40 % vprašanih uporabnikov produkta trdi, da produkt enostavno morajo imeti oziroma bi bili zelo razočarani, če produkta ne bi mogli več uporabljati. [18]

## 2.3 Agilen razvoj

Modeli razvoja programske opreme so se skozi leta razvijali od izjemno togih do vedno bolj prilagodljivih in učinkovitih. Če upoštevamo bistvene dejavnike vitkega podjetništva, lahko modele razvoja razdelimo v tri skupine [19]:

- Tradicionalni razvoj produkta (kaskadni): problem je znan, rešitev je znana; enota napredka = naslednja faza projekta.
- Sodobni razvoj produkta (agilni): problem je znan, rešitev je neznana; enota napredka = vrstica delujoče kode.
- Vitki razvoj produkta (agilni + razvoj kupcev): problem je neznan, rešitev je neznana; enota napredka = validirano učenje.

Agilne metode v primerjavi s tradicionalnimi nudijo večjo transparentnost in prilagodljivost, produkti mnogo hitreje lahko dosežejo višjo poslovno vrednost (oz. dosežejo nivo NSP), poleg tega pa drastično znižajo tveganje za razvoj neuporabnega produkta (slika 2.5).




Slika 2.5: Primerjava agilnih in tradicionalnih modelov razvoja. [20]

Eric Ries priporoča uporabo metodologije Kanban, katere proces dopolnimo z validiranim učenjem [21]. Od vseh metod je Kanban najbolj osredotočen na učinkovit razvoj, zato se lepo ujame z vitkim podjetništvom. Nekatere izmed glavnih lastnosti Kanban pristopa so [22]:

- Vizualizacija poteka dela (angl. *visualize the workflow*): Z vizualizacijo zagotovimo, da vsi udeleženi razumejo proces dela. Za vizualizacijo običajno uporabimo tablo s stolpci, ki predstavljajo različne stopnje v procesu, na stolpce pa lepimo samolepilne lističe, ki predstavljajo posamezne naloge.

- Omejevanje razvoja (angl. *limit work-in-process*): Zahtevkom določimo prioriteto in posledično lahko izločimo tiste, ki imajo najnižjo prioriteto, da z njimi ne izgubljamo časa.

- Upravljanje poteka (angl. *manage flow*): Zavedati se moramo, kako vrednost produkta potuje skozi cikel razvoja ter kje porabljamo največ virov. Potem lahko v vsakem ciklu uvedemo spremembe, ki izboljšajo efektivno porabo sredstev.

- Jasne politike (angl. *make policies explicit*): Celotna ekipa, ki sodeluje pri razvoju, mora razumeti vse procese razvoja.

- Uvedba zank za povratne informacije (angl. *implement feedback loops*): Kanbanov model spodbuja konstantne majhne izboljšave, ki so doprinos celotne ekipe.

Kanban tabla lahko vsebuje različne stolpce, odvisno od potreb posamezne ekipe, so pa v procesu vitkega podjetja običajno potrebne vsaj štirje stopnje:

- Zaloga (angl. *backlog*): Vse nove lastnosti ali izboljšave obstoječih lastnosti gredo najprej v ta stolpec, kjer jih razporedimo po prioriteti. Vsaka lastnost bi morala biti v velikosti minimalne tržne lastnosti (angl. *minimal marketable feature – MMF*).

- V teku (angl. *in progress*): Najvišjo lastnost iz zaloge prenesemo v ta stolpec. Idealno ima vsak član ekipe v teku le eno lastnost.

- Končano (angl. *built*): Ko lastnost prenesemo v ta stolpec, to pomeni, da je razvoj zaključen, lastnost je bila implementirana v produkt in že ponuja validirano učenje.

- Validirano (angl. *validated*): Ko smo na podlagi učenja ugotovili, da je lastnost zares uporabna, jo prestavimo v ta stolpec. V nasprotnem primeru, če lastnost ni bila validirana, jo odstranimo iz produkta.

# Poglavje 3

# Predstavitev problema in rešitve

Ideja za produkt se je porodila iz lastne potrebe. Za osebne, študijske in službene potrebe smo brali veliko člankov na spletu. Od tega smo si zapomnili manj, kot bi si želeli, zato smo pričeli razmišljati o rešitvi.

## 3.1 Problem: informacijska onesnaženost

Internet ponuja ogromno količino brezplačnega znanja, a branje na spletu lahko uporabniku predstavlja nekaj izzivov. Danes smo v vsakdanjem življenju, na spletu pa še posebej, neprestano obkroženi z ogromno količino informacij. Povprečen Američan je bil v letu 2008 dnevno izpostavljen 100.500 besedam in 34 GB ostalih informacij. Tretjino teh besed in več kot polovico ostalih informacij so osebe konzumirale preko računalnika. Branje, ki je bilo zaradi popularnosti televizije sicer v zatonu, pa se je po zaslugi računalnikov in interneta med leti 1980 in 2008 potrojilo. [23]

Naši možgani na takšne količine informacij niso navajeni, poleg tega pa smo na računalnikih obkroženi še s številnimi drugimi dejavniki, ki nam zmanjšujejo koncentracijo (kot so naprimer opomniki o novi elektronski pošti). Zato pogosto pozabimo podatke, ki se nam sicer zdijo pomembni, saj naši

možgani prioritizirajo podatke, ki smo jih prejeli pred nedavnim, ostale pa lahko hitro pozabijo.

Dodaten problem, s katerim se na internetu srečujemo zaradi informacijske onesnaženosti, je velika količina vsebin, ki jih želimo predelati, a za to nimamo zadostnega časa. Posledice so običajno velike količine odprtih zavihkov v spletnem brskalniku, veliko število zaznamkov ali dolgi seznami povezav do spletnih strani, ki jih želimo prebrati.

## 3.2 Učenje, pomnjenje in ponavljanje v presledkih

Poenostavljeno lahko rečemo, da naš spomin v možganih predstavljajo nevroni in sinapse. Nevroni so delci v možganih, ki pošiljajo in sprejemajo električne signale, sinapse pa so povezave med nevroni. Ko prikličemo nek spomin, se sproži veriga signalov po sinapsah med določenimi nevroni – ta specifična veriga predstavlja nek določen spomin. Jakost signalov, ki se pretakajo po sinapsah, je odvisna od tega, kako pogosto je sinapsa v uporabi – pogosteje ko jo uporabljamo, večja bo jakost signala in posledično boljši spomin. [24]

Signale med sinapsami torej lahko ojačamo tako, da jih čim pogosteje uporabljamo. To lahko dosežemo s ponavljajočim vnosom istih podatkov (pasivno učenje). Ta postopek je mnogo bolj učinkovit, če možganom podatka pri ponavljanju ne postrežemo, temveč jih prisilimo, da podatek skušajo sami priklicati iz spomina (aktivno učenje). Na ta način se sinapse precej močneje ojačajo. [25]

Če iste podatke zaporedoma vnašamo v možgane večkrat v krajšem obdobju (npr. študentje pred izpitom), bodo ti podatki v možganih ostali shranjeni relativno kratek čas. Če pa jih vnašamo v vedno daljših presledkih, možgani zaznajo, da so ti podatki potrebni tudi za daljše obdobje, zato jih v spominu ohranijo bistveno dlje časa. [26]

Aktiven priklic podatkov iz spomina torej poveča število informacij, ki

si jih zapomnimo, saj hitreje ojača sinapse. Ponavljanje po presledkih pa naredi informacije dosegljive dlje časa, saj upočasni slabenje sinaps.

## 3.3 Upravljanje s časom

Če se učimo iz knjige, lahko relativno hitro izračunamo, koliko časa bo učenje trajalo glede na število strani in našo hitrost branja. Vseeno se lahko zaradi pomanjkanja koncentracije in nepričakovanih prekinitev čas drastično poveča. To je še bolj verjetno, če si časa vnaprej ne izračunamo in splaniramo. Ko si zadamo neko časovno omejitev, je namreč mnogo bolj verjetno, da se bomo lažje osredotočili in se izognili motnjam oziroma jih ignorirali.

Če nam je predvideni čas za branje določenega besedila znan, nam to služi kot nekakšen cilj, ki ga moramo doseči. Poleg tega nam poznavanje predvidenega časa branja omogoča, da to branje umestimo v svoj urnik oziroma urnik prilagodimo času branja, ki ga moramo opraviti.

V primeru branja na internetu je ta postopek precej otežen. V kolikor si shranimo večje število člankov, iz katerih se želimo učiti, je ugotavljanje predvidenega časa branja skoraj nemogoče. S tem je oteženo tudi planiranje našega časa (kdaj bomo kaj prebrali), postavljanje časovnih okvirjev (v kolikšnem času moramo predelati določeno količino materiala) in ciljev (koliko bomo prebrali do določene točke v času).

## 3.4 Rešitev

Z našim produktom smo skušali nasloviti prej naštete probleme in tako povečati učinkovitost učenja z branjem člankov na internetu.

Članke, ki jih želimo prebrati, je izjemno enostavno shranjevati v spletno aplikacijo. Aplikacija prešteje število besed v članku in na podlagi tega izračuna predviden čas branja. Aplikacija tako služi kot seznam materialov, ki jih želimo predelati, obenem pa nam izračuna, koliko časa bomo porabili za posamezen članek. Tako nam omogoča alociranje časa za branje oziroma


Slika 3.1: Primerjava branja istega članka v naši aplikaciji (levo) in na spletnem portalu rtvslo.si (desno).

hitro iskanje kratkih člankov, v primerih, ko želimo z branjem zapolniti le kratke periode neizkoriščenega časa.

Med branjem aplikacija omogoča enostavno označevanje besedila, ki se nam zdi bistvenega pomena in si ga želimo zapomniti. Označeno besedilo enostavno shranimo, aplikacija pa poskrbi, da v primernih (vedno daljših) časovnih razmakih dobimo opomnike, da si to besedilo ponovno preberemo.

Shranjene članke lahko beremo neposredno v aplikaciji, ki ponuja prijetno bralno izkušnjo z velikimi črkami in lahko berljivo tipografijo, brez oglasov, menijev, ikon in ostalih motečih elementov (slika 3.1).

# Poglavje 4

# Razvojna orodja in tehnologije

Bistvo metodologije vitki startup je čim hitrejše učenje, zato mora tudi razvoj potekati izjemno hitro, četudi se moramo zaradi tega včasih zadovoljiti z rešitvijo, ki ni idealna. Pri razvoju je zato priporočljiva uporaba že razvitih ogrodij in primerov dobre prakse, ki pospešijo delo.

## 4.1 Heroku

V začetku razvoja, ko ne pričakujemo veliko uporabnikov in težke obremenitve strežnikov, se lahko zanesemo na kakršnokoli gostovanje. V primeru visoke izpostavljenosti produkta in posledične obremenitve strežnikov pa se lahko situacija hitro spremeni in gostovanje moramo nadgraditi z boljšo konfiguracijo in običajno dražjim plačilnim načrtom. Nadgradnja iz nizko zmogljivega gostovanja na boljši sistem je lahko dolgotrajna, časovno potratna in draga.

Heroku je oblačna platforma kot storitev (angl. *Platform as a Service – PaaS*), kjer lahko hitro začnemo z delom in se izognemo problematični nadgradnji na boljšo infrastrukturo. Heroku ponuja zanesljivo okolje za gostovanje spletnih aplikacij, kamor po preprosti začetni vzpostavitvi enostavno potisnemo (angl. *push*) izvorno kodo aplikacije preko Git repozitorija. Tako se izognemo dolgotrajnemu postopnemu vzpostavljanju okolja in se namesto

tega osredotočimo na razvoj aplikacije.

Heroku nudi podporo za večino programskih in skriptnih jezikov, ki jih danes uporabljamo za razvoj spletnih aplikacij:

- Ruby
- Python
- Node.js
- PHP
- Java
- Scala
- Clojure
- Perl

Enota računske moči v okolju Heroku se imenuje Dyno. Za osnovne potrebe zadostuje že en Dyno. Kasneje pa lahko v primeru hitre rasti našega produkta in posledično potrebe po več sredstvih število Dynov povečamo zgolj z vpisom enega ukaza v ukazno vrstico. Prednost je tudi v tem, da je uporaba enega Dyna brezplačna.

Uporabna lastnost platforme Heroku so tudi dodatki (angl. *add-ons*). Ti omogočajo enostavno dodajanje raznih strežniških aplikacij, s katerimi lahko na primer delamo varnostne kopije, spremljamo dnevnike (angl. *log*), pošiljamo elektronska sporočila iz naše aplikacije, izvajamo opravila v ozadju, procesiramo slike ali video, implementiramo SSL certifikate itd. Večino teh dodatkov lahko z omejenimi zmogljivostmi uporabljamo brezplačno; če potrebujemo večjo zmogljivost in napredne funkcije, pa jih lahko enostavno nadgradimo preko uporabniškega vmesnika ali ukazne vrstice.

Med dodatke spada tudi Heroku Postgres [27] – to je podatkovna baza kot storitev (angl. *Database as a Service*), na kateri teče PostgreSQL. To je enostavna rešitev, ki jo v Heroku okolju uporabljamo za podatkovno bazo. Z

bazo lahko upravljamo preko grafičnega spletnega vmesnika ali preko ukazne vrstice. [28]

## 4.2 Python in Django

Tudi pri izbiri programskega jezika za strežniško stran smo se osredotočili na hiter in agilen razvoj, zato so na izbiro vplivali predvsem naslednji dejavniki:

- Produktivnost jezika: enostavnost sintakse, fleksibilnost programskih pristopov, obsežnost standardne knjižnice.
- Skupnost: se bomo lahko v primeru težav zanesli na odgovore skupnosti in morda uporabili že razvite module ali bomo morali vse rešitve odkriti sami?
- Trenutno znanje: kako hitro lahko začnemo z razvojem in koliko časa bomo morali ob tem posvetiti priučevanju?

Po teh kriterijih je bila izbira dokaj preprosta. Python je skriptni jezik, ki sicer ne slovi po izjemno hitrem izvajanju, omogoča pa zelo hitro in produktivno razvijanje, saj je bil razvit prav z namenom produktivnosti in enostavno berljive kode.

V kombinaciji s Pythonom smo se odločili tudi za uporabo ogrodja (angl. *framework*) Django, s katerim lahko razvoj še bolj poenostavimo in pospešimo, obenem pa uporabljamo dobre prakse, ki so že implementirane v ogrodju. Nekaj koristnih funkcionalnosti ogrodja Django:

- ORM (Object-Relational Mapping): zelo enostavno in zmogljivo delo s podatkovno bazo.
- Obrazci (angl. *forms*): poenostavljena izdelava obrazcev, validacija vnešenih podatkov, čiščenje prejetih podatkov ter vnos v podatkovno bazo.

- Avtentikacija: enostavna in varna registracija ter avtentikacija uporabnikov.
- Administracija: prilagodljiv pregled in upravljanje uporabniških računov ter ostalih vsebin, ki jih hranimo v podatkovni bazi.
- Predloge (angl. *templates*): izjemno zmogljive in prilagodljive predloge, preko katerih dinamično zgradimo HTML strani.
- Seje (angl. *sessions*): enostavno delo z uporabniškimi sejami.

## 4.3 HTML5, CSS3, jQuery

Vsaka spletna stran temelji na označevalnem jeziku HTML, ki predstavlja strukturo spletne strani. Danes praktično vse spletne strani uporabljajo tudi CSS, ki poskrbi za obliko in stil, ter JavaScript, ki strani doda interaktivnost in površinsko funkcionalnost. V našem produktu smo se poslužili nekaterih funkcionalnosti omenjenih jezikov, ki so stopile v veljavo šele pred kratkim.

Primeri uporabe HTML5:

- Semantični elementi (`header, section, footer, nav,...`): omogočajo lažje berljivo strukturo HTML dokumenta.
- Privzeta vrednost vnosnega polja v obrazcu (`value=‘‘Ime’’`): prikaže bledo vrednost v vnosnem polju, ki izgine, ko začnemo tipkati.
- Avtomatski fokus na vnosnem polju v obrazcu (`autofocus`): takoj, ko se stran naloži, bosta fokus in kazalka v izbranem vnosnem polju.

Primeri uporabe CSS3:

- Animacija (`animation`): postopna sprememba elementa iz enega v drug stil.
- Tranzicija (`transition`): postopna sprememba elementa iz enega v drug stil, ki ima vedno neko začetno in končno stanje.

- Prosojnost (`transparency`).
- Senca (`box-shadow`).
- Uporaba nenameščenih pisav (`@font-face`).

Uporabili smo tudi knjižnico jQuery, ki olajša in pospeši delo z JavaScriptom. jQuery omogoča enostavno izbiranje in manipuliranje z DOM elementi, animacije, ravnanje z dogodki (angl. *event handling*), ustvarjanje Ajax klicev ipd.

## 4.4 Readability API in Pocket API

Za delovanje naše aplikacije smo se zanesli tudi na uporabo dveh APIjev. Razlog za uporabo APIjev je ta, da se lahko zanesemo na algoritme in aplikacije, ki so že dlje časa v uporabi, zato je njihova zanesljivost in uporabnost na visokem nivoju. Razvoj lastne rešitve bi namreč zahteval veliko sredstev, torej bi bilo to zelo potratno dejanje, dokler ni potrjeno, da je produkt zanimiv za trg oz. dokler nimamo zelo specifičnih zahtev, zaradi katerih bi potrebovali lastno rešitev.

Readability API omogoča zanesljivo razčlenjevanje vsebine spletnih strani (angl. *content parsing*). APIju podamo le URL članka, v odgovoru pa dobimo prečiščen tekst članka (brez nepotrebnih elementov in stilov, ki jih sicer vsebuje spletna stran), naslov članka, število besed ter še nekaj podatkov, ki za naš produkt niso pomembni.

Tako prečiščen članek lahko v aplikaciji prikažemo z našim prirejenim stilom in ustvarimo prijetno okolje za branje. Število besed pa uporabimo za izračun, koliko časa bo uporabnik porabil za branje posameznega članka.

Pocket (prvotno imenovan *Read It Later* – Preberi kasneje) je namenjen enostavnemu shranjevanju spletne vsebine za kasnejše branje. Za nas najbolj pomembna lastnost aplikacije Pocket je ta, da je poleg spletne aplikacije in dodatka za brskalnike (Google Chrome, Firefox, Safari) dosegljiva tudi na veliki večini najpogosteje uporabljenih mobilnih in namiznih operacijskih

sistemov – OS X, Windows, iOS, Android, Windows Phone, BlackBerry ter Kobo Touch.

Pocket torej uporabniku omogoča, da ne glede na to, kje je in katero napravo uporablja, v vsakem trenutku lahko vsebine, ki jih je našel na internetu, enostavno in hitro shrani za kasnejši pregled. Preko Pocket APIja lahko te vsebine za vsakega uporabnika prenesemo v svojo aplikacijo. V našem primeru torej uporabnik lahko poveže naš produkt s svojim Pocket računom in vse vsebine, ki jih shranjuje v Pocket, se bodo avtomatsko pričele shranjevati tudi v našo aplikacijo.

# Poglavje 5

# Iteracije poslovnih okvirjev in razvoj kupcev

Od prvotne do končne ideje produkta, ki se je izkazala kot zanimiva za trg, smo naredili tri iteracije. Vmes smo se pogovarjali s strankami in zgradili različne NSPje.

## 5.1 Prva iteracija

Prvotna ideja za produkt je temeljila predvsem na dejstvu, da osebam, ki veliko berejo na internetu, povzroča frustracije to, da ne preberejo vsega, kar želijo. Tipični uporabniki so uporabniki aplikacije Pocket, saj je zelo enostavno prekomerno shranjevanje vsebine v Pocket, medtem pa se sploh ne zavedamo, koliko branja smo si že nabrali na zalogo.

Pocket je že uveljavljena aplikacija. Po besedah ustanovitelja Natea Weinerja so v letu 2014 imeli 11 milijonov uporabnikov [29]. Zaradi tega se nam je zdelo smiselno, da bi se sprva osredotočili kar na uporabnike aplikacije Pocket, katere moti, da ne preberejo vsega, kar shranijo v Pocket.

### 5.1.1 Poslovni okvir

Poslovni okvir je v prvi iteraciji vseboval sledeče hipoteze:

- Segmentacija kupcev:
  - Uporabniki aplikacije Pocket, katere moti, da ne preberejo vsega kar shranijo v Pocket.
- Odnosi s kupci:
  - Elektronska pošta
  - Socialna omrežja – Facebook, Twitter
- Distribucijske poti:
  - Forumi, kjer se zadržujejo ljudje, ki veliko berejo na internetu.
  - Facebook skupine, kjer se zadržujejo ljudje, ki veliko berejo na internetu.
  - Lastni blog: objavljanje nasvetov o hitrejšem in bolj efektivnem branju ipd.
  - Objave člankov na drugih relevantnih blogih
  - Objave v medijih
  - AdWods in Facebook ciljno oglaševanje
- Ponudba vrednosti:
  - Uporabniki bodo dejansko prebrali shranjene članke.
  - Uporabniki bodo vsak dan postajali pametnejši in boljše različice sebe.
- Ključne aktivnosti:
  - Razvoj aplikacije
  - Pisanje blog objav in ostalih objav za medije
  - Iskanje partnerjev za promocijo
- Ključni viri:

  - Strežnik
  - Domena

- Ključni partnerji:
  - Pocket
  - Pisci blogov
- Struktura stroškov:
  - Strežnik in domena
  - AdWords in Facebook oglasi
- Viri prihodkov:
  - Premium uporabniki – mesečna/letna naročnina na plačljiv model brez omejitev

### 5.1.2 Intervjuji s strankami

Prve potencialne stranke za intervjuje smo iskali med svojimi znanci ter preko socialnih omrežij. Intervjuje smo imeli s 27 posamezniki, ki so uporabniki aplikacije Pocket. Cilj intervjujev je bil čim bolje spoznati uporabniške bralne navade ter frustracije, ki jih imajo ob tem; predvsem pa ugotoviti, če jih zares moti, da ne preberejo vsega, kar shranijo. Vprašanja smo prilagajali vsakemu posamezniku, nekatera izmed ključnih vprašanj pa so bila:

- kaj konkretno berejo na internetu;
- zakaj prebirajo vsebine, ki jih prebirajo (učenje, zabava,...);
- kolikšen delež člankov, ki jih shranijo v Pocket, kasneje zares preberejo;
- kje običajno članke shranjujejo v Pocket (preko mobilnih ali namiznih naprav);
- kje kasneje prebirajo članke (na mobilnih ali namiznih napravah);

- kako se odločijo, katere članke preberejo najprej;
- ali jih moti, da ne preberejo vsega, kar shranijo v Pocket.

Najpomembnejša ugotovitev je bila, da večine uporabnikov ne moti, da ne preberejo vsega. Čeprav so intervjuvanci v večini rekli, da preberejo le 20-50 % shranjenih člankov, jih preostale vsebine kasneje ne zanimajo več toliko oziroma dobijo veliko količino novega materiala, ki jim je kasneje pomembnejši. Z intervjuji smo torej ovrgli hipotezo o našem segmentu kupcev.

## 5.2 Druga iteracija

V drugi iteraciji smo spremenili hipotezo o kupcih. Sedaj smo predpostavili, da si uporabniki aplikacije Pocket stvari, ki jih preberejo, ne zapomnijo dovolj dobro in jim to povzroča frustracije.

### 5.2.1 Poslovni okvir

Večina poslovnega okvira je ostala enaka, spremenili smo le dve polji:

- Segmentacija kupcev:
  - Uporabniki aplikacije Pocket, katere moti, da si ne zapomnijo vsega, kar preberejo.
- Ponudba vrednosti:
  - Uporabniki si bodo bolje zapomnili, kar so prebrali v člankih, ki so jih shranili v Pocket.
  - Uporabniki bodo vsak dan postajali pametnejši in boljše različice sebe.

### 5.2.2 Intervjuji s strankami

Izmed oseb, ki so sodelovale v prvem intervjuju, smo jih sedaj ponovno intervjuvali 23. Preostali posamezniki so se v prvem intervjuju izkazali za neprimerne, ker so Pocket uporabljali v izjemno majhnem obsegu ali na precej neklasičen način.

Tokrat smo se osredotočili na sledeča vprašanja:

- koliko vsebin, ki jih preberejo na internetu, si dejansko zapomnijo;
- ali jih moti, da si ne zapomnijo vsega;
- katere vrste vsebin menijo, da si zapomnijo bolje;
- kako se trenutno lotevajo rešitve, da bi si zapomnili več.

Ugotovili smo, da skoraj polovice uporabnikov ne moti, da si ne zapomnijo vsega. Več kot polovica pa je vseeno priznala, da jih to vsaj delno moti. Nekateri intervjuvanci so povedali tudi, da si delajo zapiske ali pa članke, ki se jim zdijo izjemno pomembni, ponovno preberejo.

To se nam je zdel zadosten delež za potrditev hipoteze o strankah in nadaljevanje v tej smeri.

### 5.2.3 Razvoj prototipa

Z raziskovanjem efektivnih metod učenja smo dobili idejo za produkt s ponavljanjem v presledkih. Da bi si končni produkt lažje predstavljali in definirali vse procese ter preverili, če metoda zares deluje, smo želeli najprej sestaviti prototip in ga testirati v kontroliranem okolju na lastni koži. Za prototip smo uporabili program Evernote, ki je v prvi vrsti namenjen beleženju in shranjevanju zapiskov.

Preko storitve IFTTT (https://ifttt.com/), ki omogoča medsebojno povezovanje številnih spletnih storitev, smo povezali Pocket in Evernote, tako da so naslovi in povezave do vseh člankov, ki smo jih shranili v Pocket, avtomatsko pristali označeni z določeno oznako (angl. *tag*) v določeni mapi v

programu Evernote. V tej mapi smo hranili vse članke, oznaka "Za branje" pa je označevala tiste članke, ki jih še nismo prebrali.

Za presledke med ponovitvami branja smo uporabili intervale 2 dneva, 8 dni, 20 dni in 30 dni. Da bi spremljali, kdaj moramo prebrati določene izpiske, smo uporabili Evernote funkcijo opomnikov. Vsak zapis v Evernote namreč lahko spremenimo v opomnik ter mu dodamo datum, kdaj želimo dobiti opomnik.

V Evernote smo ustvarili še šest oznak – za vsako iteracijo branja svojo ter dodatno oznako, ki je povezovala vse članke, ki smo jih že prebrali. Struktura oznak je vsebovala sledeče elemente:

- Za branje (članki, ki jih še nismo prebrali)
- Prebrano (vsi članki, ki smo jih že prebrali, od prve do zadnje iteracije)
- Iteracija 1 (izpiski iz člankov po prvem branju člankov)
- Iteracija 2 (izpiski iz člankov po drugem branju oz. prvem branju izpiskov)
- Iteracija 3 (izpiski iz člankov po tretjem branju oz. drugem branju izpiskov)
- Iteracija 4 (izpiski iz člankov po četrtem branju oz. tretjem branju izpiskov)
- Arhiv (izpiski iz člankov po petem branju oz. četrtem branju izpiskov)

Ker smo še vedno želeli testirati tudi uporabnost poznavanja dolžine branja posameznega članka, smo vsak večer v brskalniku odprli vse nove članke, jih z Readability dodatkom za Chrome očistili vseh elementov razen teksta, nato pa uporabili Spreeder (http://www.spreeder.com/app.php) aktivni zaznamek (angl. *bookmarklet*), ki prešteje število besed, ki smo jih označili na spletni strani. Tako smo na relativno hiter način prišli do števila besed posameznega članka. Tega smo delili s 150 (povprečna hitrost branja v besedah na minuto [30]) ter število minut zapisali poleg naslova članka v Evernote.

Procesiranje posameznega članka je torej potekalo po naslednjih korakih:

1. Članek iz katere koli naprave dodamo v Pocket.
2. Preko storitve IFTTT članek avtomatsko pristane v Evernote označen z oznako "Za branje".
3. Zvečer za članek izmerimo dolžino branja in podatek dodamo k naslovu v Evernote.
4. Ko imamo čas, članek preberemo in si ob tem pomembne odseke po principu kopiraj-prilepi shranimo v Evernote zapis, ki pripada temu članku.
5. Ko končamo z branjem, zapisu odstranimo oznako "Za branje", namesto nje pa dodamo oznaki "Prebrano" ter "Iteracija 1".
6. Zapis spremenimo v opomnik z datumom dva dneva kasneje od datuma, ko smo članek prebrali.
7. Po dveh dnevih dobimo opomnik – tokrat preberemo le shranjene odseke članka, odstranimo oznako "Iteracija 1" in jo nadomestimo z "Iteracija 2" ter prestavimo opomnik za 8 dni kasneje.
8. Korak 7 ponavljamo za vseh pet iteracij, zatem pa zapisu dodamo oznako "Arhiv" in odstranimo opomnik.

Ob začetku uporabe prototipa smo nekaj člankov prebrali in jih shranili, a z njimi nismo šli čez celoten proces – te članke smo uporabili kot kontrolno skupino, da bi primerjali, koliko smo si zapomnili v primerjavi s članki, ki bodo šli čez celoten proces 5 iteracij ponavljanja.

Po 60 dneh uporabe prototipa so prvi članki šli preko vseh iteracij (2 dneva + 8 dni + 20 dni + 30 dni). Počakali smo še dva tedna, da se je nabralo več člankov in se je znanje prvih člankov nekoliko poleglo po zadnji iteraciji. Potem smo naredili primerjavo med znanjem. Vzeli smo 10 člankov iz kontrolne skupine (članki, ki smo jih prebrali pred 60-70 dnevi)

ter 10 člankov iz testne skupine (članki, s katerimi smo šli čez vseh 5 iteracij ponavljanja).

Za vsakega izmed člankov smo na podlagi naslova morali zapisati čim več podatkov, ki smo si jih zapomnili. V primeru člankov iz kontrolne skupine si v nekaterih primerih nismo zapomnili popolnoma nič konkretne vsebine, morda le zelo grob oris, o čem je članek govoril. V najboljših primerih smo si zapomnili majhno količino specifičnih podatkov; predvsem zato, ker so na nas naredili močan vtis ali pa zato, ker smo jih v tem času že uporabili v praksi.

Rezultati preverjanja znanja iz testne skupine so bili bolj spodbudni. Pri vseh člankih smo si zapomnili večino podatkov iz zapiskov, v večini primerov pa smo si zapomnili tudi številne druge podatke iz članka.

Sklepi raziskave s prototipom so bili sledeči:

- Dolgotrajnih učinkov takšnega učenja sicer nismo izkusili, na krajše obdobje pa je tehnika v našem primeru delovala izjemno dobro.

- Takšno ponavljanje zahteva veliko količino samodiscipline, saj v primeru vsakodnevnega intenzivnega branja novih člankov lahko včasih na isti dan dobimo opomnike za veliko količino člankov, ki so v različnih fazah skozi iteracije. To lahko delno rešimo s pametnim algoritmom, ki razporedi opomnike na kakšen dan prej ali kasneje od striktnega datuma.

- Poznavanje dolžine branja nam je pomagalo pri prioritetiziranju branja novih člankov. Sicer so zaradi tega daljši članki včasih nekoliko izostali, a smo si zanje lažje vzeli čas na primer ob koncu tedna.

- Pogosto je koristno, če lahko v zapiske dodamo tudi slikovno gradivo, ne le tekst. To še dodatno pomaga pri pomnjenju, še posebno, če smo vizualni tip človeka.

- Izjemno pomembno je imeti vso funkcionalnost na enem mestu in kolikor se da poenostavljeno – zelo neudobno je namreč imeti seznam

branja v eni aplikaciji, brati v drugi, shranjevati zapiske spet v prvi, ročno nastavljati opomnike, itd.

- Dobro je imeti podatek, kdaj je bil kateri članek dodan, saj nam je tudi to včasih pomembno pri prioritiziranju branja.

- Pomembno je, da poleg zapiskov ohranimo tudi povezavo do originalnega članka. Včasih si še vedno želimo ogledati kakšne podrobnosti, ki si jih nismo shranili, ali pa originalni članek potrebujemo za referenco.

Proces uporabe prototipa je bil sicer relativno zapleten, a njegova izgradnja je bila popolnoma brezplačna in ni trajala dlje kot nekaj ur. S testiranjem smo prišli do pomembnih ugotovitev, ki smo jih upoštevali pri nadaljnjem razvoju in raziskavah.

Ker je pozabljanje najhitreše v prvih nekaj urah [31], bi v idealnem primeru prve ponovitve naredili še isti dan, kar pa je običajno nepraktično. Zato smo se odločili za nadaljnji razvoj spremeniti presledke za ponavljanje na 1 dan, 5 dni, 25 dni in 120 dni. Zadnji interval smo precej podaljšali na podlagi predpostavke, da bo to bolje vplivalo na dolgotrajni spomin.

### 5.2.4 Razvoj in lansiranje NSPja

Prototip, ki smo ga zgradili za lastno uporabo, je bil mnogo preveč zapleten, da bi ga lahko dali v testiranje ostalim uporabnikom. Zato smo se odločili, da na podlagi vsega, kar smo se naučili iz intervjujev in prototipa, razvijemo pristajalno stran in najosnovnejši sprejemljivi produkt.

NSP so sestavljale sledeče funkcionalnosti:

- Registracija in avtentikacija uporabnikov

- Osnovne uporabniške nastavitve

- Ročno dodajanje člankov

- Sinhronizacija z aplikacijo Pocket

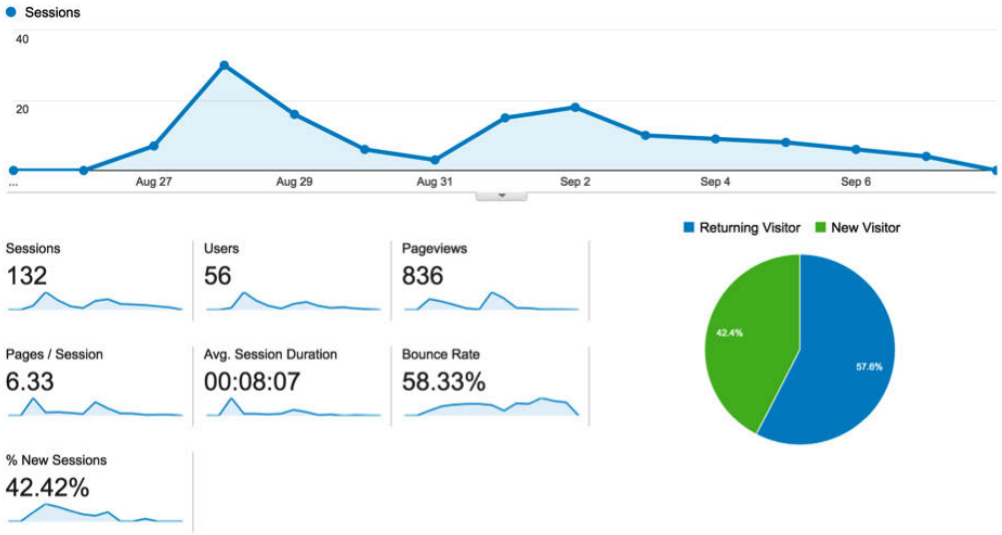
- Pregled vseh vnešenih člankov skupaj z njihovim predvidenim časom branja
- Branje posameznega članka
- Brisanje posameznega članka
- Arhiviranje posameznega članka
- Pregled vseh arhiviranih člankov
- Shranjevanje pomembnih delov članka z označevanjem besedila
- Pregled vseh člankov s shranjenimi izpiski
- Pregled vseh shranjenih izpiskov posameznega članka
- Opomniki za ponavljanje v presledkih (preko elektronske pošte)
- Možnost vklopa in izklopa opomnikov za posamezen članek

Za obliko in vizualno podobo strani smo uporabili brezplačno predlogo (angl. *template*) HTML5 UP Twenty (http://html5up.net/twenty), ki je zelo prilagodljiva in že prirejena za prikaz na mobilnih napravah.

Na pristajalno stran ter preko celotnega produkta smo implementirali Google Analytics, da smo lahko spremljali obnašanje uporabnikov.

Preden smo produkt predstavili javnosti, smo potrebovali tudi ime. Želeli smo, da je ime kratko, enostavno in povezano s pomnjenjem. Iskali smo različne variacije besed v različnih jezikih in se na koncu odločili za latinski prevod besedne zveze "jaz pomnim" – Memini [32].

Produkt smo 27. 8. 2014 objavli na spletni strani Hacker News (https://news.ycombinator.com/). Tako v naslovu objave kot tudi na pristajalni strani smo poudarili, da je produkt namenjen uporabnikom aplikacije Pocket. Naslov objave se je glasil "Preberi vse Pocket članke in si zapomni, kar je pomembno" (angl. *Read all your Pocket articles and memorize stuff that matters*).




Slika 5.1: Google Analytics po prvi objavi Memini na Hacker News.

Nekaj dni po objavi je bilo iz Google Analytics razvidno (slika 5.1), da je stran obiskalo 56 novih unikatnih obiskovalcev, od tega se jih je 6 registriralo. To predstavlja skoraj 11 % konverzijo, kar je sicer visoko, a ne preveč relevantno, saj so rezultati iz tako majhnih številk statistično zanemarljivi. Memini bi lahko objavili še kje drugje, da bi dobili več uporabnikov, a smo želeli za začetek spremljati obnašanje manjšega števila uporabnikov. Produkt smo čez nekaj časa dali v uporabo tudi osebam, s katerimi smo imeli intervjuje. Po nekaj dneh smo vse uporabnike kontaktirali in jih povprašali o njihovi uporabniški izkušnji in mnenju o aplikaciji.

V nadaljnih pogovorih z nekaterimi uporabniki smo ugotovili, da kljub temu, da naš produkt temelji na povezavi s Pocketom, to ni njegova najpomembnejša lastnost. Najpomembnejši so opomniki za ponavljanje v presledkih. Poleg tega je targetiranje in iskanje uporabnikov Pocketa zelo težko, saj nimajo spletne skupnosti, ki bi jo v začetku lahko uporabili za promocijski kanal. Ugotovili smo, da bi bilo morda pametneje, če bi se osredotočili v splošnem na ljudi, ki veliko berejo na internetu, in za njih naredili uporabo še enostavnejšo.

## 5.3 Tretja iteracija

Nova ideja je slonela na dodatku za brskalnik Google Chrome. Pocket integracija je ostala le kot dodatna funkcionalnost za vse, ki bi si to želeli. Glavna funkcionalnost aplikacije je sedaj bil Chrome dodatek – uporabnik lahko kadarkoli med prebiranjem člankov na internetu označi pomemben del besedila in ga z enim klikom shrani v Memini ter ob tem vklopi opomnike za ponavljanje v presledkih.

### 5.3.1 Poslovni okvir

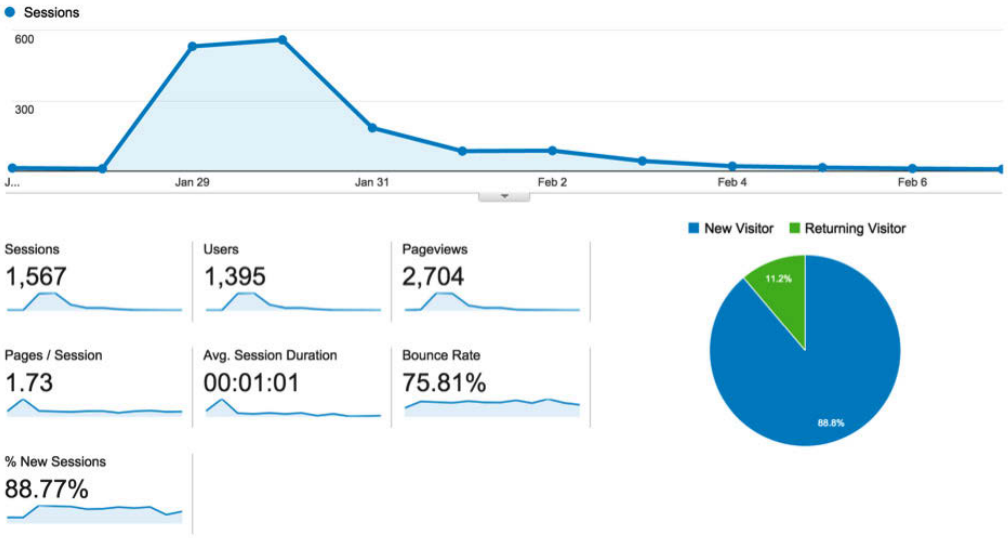
Ponovno smo morali prirediti dve polji v poslovnem okvirju, ostala polja pa so se nam še vedno zdela smiselna:

- Segmentacija kupcev:
  - Osebe, ki berejo veliko člankov na internetu in jih moti, da si ne zapomnijo vsega, kar preberejo.
- Ponudba vrednosti:
  - Uporabniki si bodo bolje zapomnili, kar so prebrali v kateremkoli članku na internetu.
  - Uporabniki bodo vsak dan postajali pametnejši in boljše različice sebe.

### 5.3.2 Razvoj in lansiranje NSPja

Glavna novost v novem NSPju je bil seveda Chrome dodatek. Ta je deloval tako, da je uporabnik na katerikoli spletni strani lahko označil del teksta in s pritiskom na gumb dodatka se je pokazalo malo okno s podatki o članku (naslov, URL ter označeno besedilo) ter potrditveno polje, kjer je uporabnik lahko izbral, če bi za ta članek želel dobivati opomnike.

Poleg odprave nekaterih hroščev in lepotnih popravkov smo ponovno posodobili tudi pristajalno stran. Ta je sedaj poudarjala učenje in pomnjenje




Slika 5.2: Google Analytics po objavi Memini na Hacker News in Product Hunt.

ter kot osnovni način uporabe ponujala Chrome dodatek. Pocket je ostal omenjen le kot sekundarna možnost za vnos člankov.

Produkt smo ponovno objavili na Hacker News, tokrat z naslovom "Enostaven in dokazan način za pomnjenje vsega, kar prebereš na internetu" (angl. *A simple and proven way to remember what you read online*). Naslov objave na Hacker News igra izjemno pomembno vlogo, saj je to edina stvar, ki jo uporabniki vidijo, ko prebirajo portal. Zgolj na podlagi naslova se torej odločijo, če je to nekaj, kar bi jih utegnilo zanimati in bodo zato obiskali spletno stran produkta. V naslovu moramo zato zelo efektivno skomunicirati ponudbo vrednosti našega produkta.

Nekaj ur po objavi je spletno stran našega produkta že obiskalo skoraj dvakrat več obiskovalcev kot pri prejšnji objavi. Očitno je bila aplikacija nekomu dovolj všeč, da jo je objavil še na strani Product Hunt (https://www.producthunt.com/), kjer je objava pritegnila še več pozornosti. Skupen rezultat je bilo skoraj 1400 novih unikatnih obiskovalcev v enem tednu (slika 5.2) ter 270 novih registriranih uporabnikov. To pomeni več kot

19 % konverzijo, kar je izredno dober rezultat.

Ker brezplačen Heroku račun ne omogoča izjemnih zmogljivosti, smo omejili število uporabnikov, ki so po registraciji lahko uporabljali aplikacijo. Le prvih 50 je imelo aktiviran račun, ostali pa so prejeli sporočilo z obrazložitvijo, da jim bomo račun aktivirali, ko bo aplikacija v kasnejši stopnji razvoja.

Prejeli smo tudi nekaj elektronskih sporočil uporabnikov, med katerimi sta izstopali predvsem dve sporočili. Prvo je poslal študent iz Bostona, ki trpi za hudo obliko motnje pomanjkanja pozornosti s hiperaktivnostjo (angl. *attention deficit hyperactivity disorder – ADHD*) in verjame, da mu bo naš produkt lahko pomagal pri učenju. Ker ni prišel med prvih 50, je prosil, če mu lahko vseeno damo dostop do aplikacije. Sporočilo nam je odprlo nov pogled na uporabo aplikacije ter potencialen nov segment strank.

Drugo sporočilo je poslal vodja pospeševalnika za izobraževalne startupe Emerge Education iz Londona (http://emerge.education/). V sporočilu je izrazil navdušenje nad našo idejo ter ponudil vključitev v njihov pospeševalni program z možnostjo investicije. To sporočilo nam je dodatno potrdilo, da bi produkt lahko bil zanimiv za trg, saj so strokovnjaki, ki se dnevno srečujejo s številnimi inovativnimi idejami, izrazili zanimanje zanj.

S tretjo iteracijo smo torej uspešno dosegli ujemanje produkta in trga na majhnem nivoju. Navidezen začetni uspeh nas seveda ne sme zavesti, saj je bil produkt predstavljen na strani, kjer se zbirajo predvsem zgodnji uporabniki (angl. *early adopters*), kar pomeni, da niso nujno naše ciljne stranke. Večino morda le zanimajo inovativne tehnologije in ideje, zato radi preizkušajo nove produkte. Vsekakor pa smo dobili nov vpogled v uporabnike in se iz tega zadosti naučili, da lahko nadaljujemo z razvojem in testiranjem hipotez.

# Poglavje 6

# Razvoj aplikacije

Razvoja smo se lotili po agilnih principih. Produkt smo razdelili na minimalne tržne lastnosti in za sledenje procesa razvoja uporabili Kanban tablo na platformi Trello (https://trello.com).

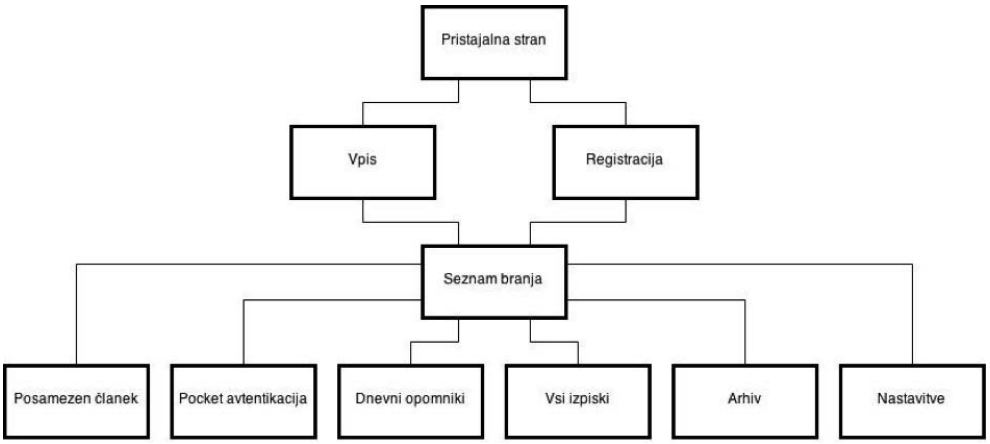
Na podlagi validiranega učenja smo ugotovili, katere funkcionalnosti mora imeti aplikacija in kakšne primere uporabe moramo upoštevati pri razvoju. Poleg funkcionalnosti, naštetih v poglavju 5.2.4, smo velik poudarek namenili enostavni uporabi in uporabniku prijazni izkušnji. Čeprav gre za NSP, se nam je zdela pozitivna uporabniška izkušnja pri lansiranju inovativnega in uporabnikom neznanega produkta zelo pomembna.

Med iskanjem poslovnega modela in kroženjem po zanki naredi-meri-spoznaj smo krožili tudi med različnimi fazami razvoja.

## 6.1 Načrtovanje

Pri načtovanju smo si veliko pomagali z belo tablo, kamor smo najprej skicirali prvo verzijo žičnih modelov (angl. *wireframe*) ter povezave med njimi. Bela tabla je zelo uporabno orodje za risanje skic in viharjenje možganov (angl. *brainstorming*), saj omogoča enostavno vizualiziranje idej za lažjo predstavo ter enostavno spreminjanje in popravljanje na podlagi novih idej.

Za skiciranje bolj podrobnih žičnih modelov smo uporabili orodje Moqups



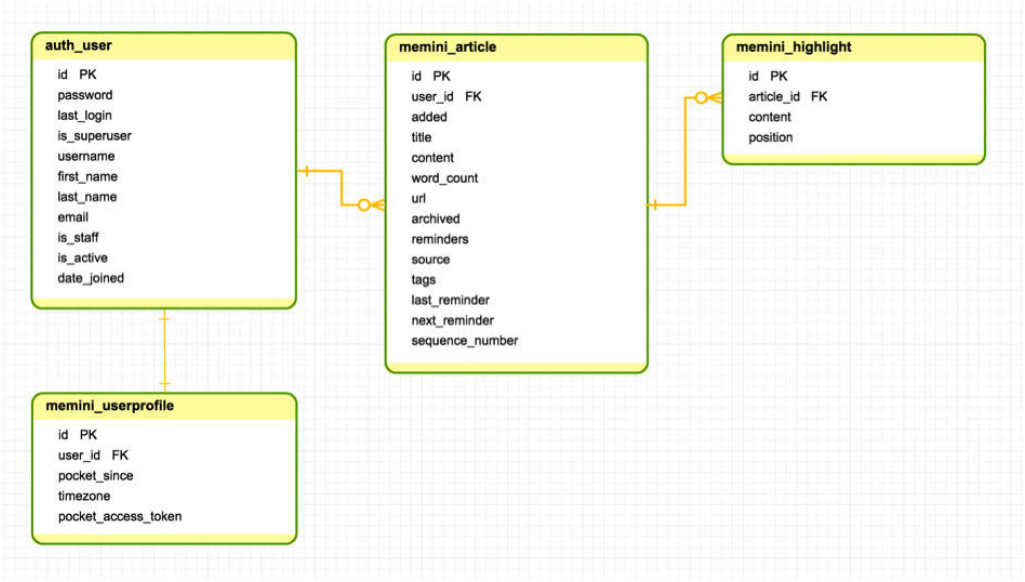

Slika 6.1: Končni načrt strani.

(https://moqups.com/). Tu smo vizualizirali idealno strukturo strani in razporeditev elementov, da smo lahko našli za naše potrebe primerno HTML predlogo (angl. *template*).

Pripravili smo načrt strani (angl. *sitemap*) (slika 6.1), ki nam je razjasnil implementacijo navigacije preko celotne aplikacije ter pokazal možne uporabniške tokove (angl. *user flow*), na katere moramo biti pozorni.

Predno smo se lotili razvoja, smo pripravili tudi načrt podatkovne baze. V ta namen smo pripravili ER diagram (slika 6.2), ki nazorno pokaže tabele ter razmerja med njimi. V diagram podatkovne baze smo zajeli vse podatke, ki jih potrebujemo za delovanje aplikacije, pri tem pa smo poskrbeli za učinkovito pridobivanje podatkov ter enostavne kasnejše dopolnitve podatkovne baze.

Tabela `auth_user` predstavlja del Django avtentikacijskega modula, ostale tabele pa smo zasnovali sami. Django poleg tega za delovanje uporablja še nekatere druge tabele, ki jih v ER diagram nismo vključili, saj niso bistvene za delovanje naše aplikacije.




Slika 6.2: Končni ER diagram podatkovne baze.

## 6.2 Implementacija

Končni produkt je bil sestavljen iz dveh ločenih celot: spletne aplikacije ter dodatka za brskalnik Google Chrome.

Spletno aplikacijo smo razvili po principu Django MVC (angl. *Model-View-Controller* – Model-Pogled-Krmilnik) arhitekture. V Django terminologiji zaradi drugačnega poimenovanja posameznih komponent tej arhitekturi včasih rečemo tudi MTV (angl. *Model-Template-View* – Model-Predloga-Pogled).

Model predstavlja implementacijo in interakcijo s podatkovno bazo. Vsebuje predvsem definicijo posameznih tabel ter osnovne funkcije, ki jih uporabljamo za delo z bazo. V modelu tudi nastavimo nekatere privzete vrednosti, ki naj jih imajo novi vnosi v podatkovni bazi.

Pogled (v Django terminologiji sicer imenovan predloga (angl. *template*)) definira, kako bodo podatki predstavljeni v brskalniku. Vsebuje predvsem HTML, CSS in JavaScript kodo, ki skupaj definirajo vizualno strukturo in izgled spletne aplikacije. Django modul za predloge omogoča združevanje

večih predlog, zato smo za našo aplikacijo pripravili dve glavni predlogi, ki smo ju za potrebe posamezne strani dopolnjevali z ostalimi predlogami. Prva glavna predloga je služila za pristajalno stran, ki ima precej drugačno strukturo od preostale aplikacije. Druga glavna predloga pa je služila kot okvir za preostali del aplikacije. Vsebovala je večino elementov spletne aplikacije (glavo, navigacijo, nogo itd.), za glavni vsebinski del pa smo pripravili posamezne ločene predloge za vsak različen pogled.

Krmilnik (v Django terminologiji sicer imenovan pogled (angl. *view*)) vsebuje večino funkcij, ki izvršujejo delovanje aplikacije, določa, kateri podatki bodo vidni uporabniku v pogledu ter preko modelov upravlja s podatkovno bazo. Krmilnike v naši aplikaciji predstavlja naslednjih 24 funkcij:

- `index` – avtenticirane uporabnike preusmeri na domačo stran, ostalim pa prikaže pristajalno stran;
- `home` – priskrbi vse podatke in omogoči delovanje za domačo stran. To zajema seznam branja, izpiske, ki jih mora uporabnik ponoviti na današnji dan, ter ročno dodajanje člankov;
- `article` – poskrbi za prikaz posameznega članka;
- `article_parse` – s pomočjo Readability APIja prikaže vsebino članka na podlagi podanega URLja (funkcija v trenutni različici aplikacije ni v uporabi);
- `register` – poskrbi za celoten proces registracije novega uporabnika s pomočjo Django modula za avtentikacijo;
- `user_login` – avtentikacija registriranih uporabnikov s pomočjo Django modula za avtentikacijo;
- `user_logout` – izpiše uporabnika in ga preusmeri na pristajalno stran;
- `pocket_oauth` – prvi korak avtentikacije s Pocket APIjem;

- `pocket_oauth_done` – drugi korak avtentikacije s Pocket APIjem ter sinhronizacija prvih 20 člankov;
- `readability_oauth` – prvi korak avtentikacije z Readability APIjem;
- `readability_oauth_done` – drugi korak avtentikacije z Readability APIjem;
- `save_highlight` – shrani izbrani tekst posameznega članka v podatkovno bazo;
- `highlights` – prikliče iz baze vse izpiske za določen članek;
- `archive_article` – označi članek kot arhiviran, kar pomeni, da ga lahko vidimo le še v arhivu;
- `unarchive_article` – obratno od prejšnje funkcije, odstrani oznako arhiviranega članka;
- `delete_article` – izbriše članek iz podatkovne baze;
- `delete_highlight` – izbriše posamezen izpisek iz podatkovne baze;
- `archive` – iz podatkovne baze prikliče članke, ki so označeni kot arhivirani, ter jih prikaže v pogledu za arhiv;
- `user_settings` – prikaz in funkcionalnost osnovnih uporabniških nastavitev (sprememba uporabniškega imena, gesla in elektronske pošte);
- `memorize` – vključi opomnike za izbrani članek (več v poglavju 6.2.1);
- `unmemorize` – obratno od prejšnje funkcije, izključi opomnike za izbrani članek;
- `daily_digest` – iz podatkovne baze prikliče in prikaže vse povzetke, katere mora uporabnik prebrati na današnji dan, glede na urnik ponavljanja v presledkih;

- `all_highlights` – iz podatkovne baze prikliče in prikaže vse povzetke, združene glede na članek, kateremu pripadajo;

- `save_from_chrome` – omogoča shranjevanje preko Chrome dodatka (več v poglavju 6.2.2).

### 6.2.1 Opomniki za ponavljanje v presledkih

Ko v aplikaciji vključimo opomnike za določen članek, temu članku v bazi določimo naslednje vrednosti:

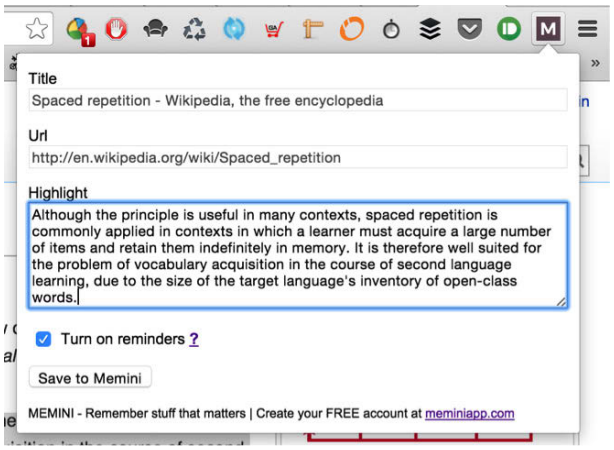
```
reminders = True
sequence_number = 1
next_reminder = date.today() + timedelta(days=1)
```

Vrednost `reminders` določa, če so opomniki za ta članek omogočeni, `sequence_number` vsebuje cikel ponavljanja za ta članek, `next_reminder` pa vsebuje datum naslednje ponovitve.

Vsako uro v ozadju aplikacije izvršimo načrtovano opravilo (angl. *scheduled task*) `send_reminders`, ki iz baze pokliče vse članke, ki imajo vključene opomnike ter vrednost `next_reminder` nastavljeno na trenutni datum. Članke združi po uporabnikih ter vsakemu uporabniku pošlje elektronsko sporočilo s povezavami do opomnikov posameznih člankov, ki jih mora uporabnik prebrati. Ob tem za vsak članek tudi poveča `sequence_number` ter spremeni vrednost `next_reminder` na datum, ko mora uporabnik naslednjič prebrati povzetke tega članka. Te periode so fiksno nastavljene na 1 dan, 5 dni, 25 dni ter 120 dni. Po četrti ponovitvi se opomniki izklopijo.

Za izvajanje načrtovanih opravil smo uporabili Heroku dodatek Scheduler [33], s katerim enostavno definiramo, katero funkcijo želimo izvesti ter kako pogosto. Funkcionalnost je zelo podobna kron opravilom (angl. *cron jobs*) [34].

Za pošiljanje elektronskih sporočil z opomniki pa smo uporabili Heroku dodatek Mandrill [35]. Ta poleg enostavnega pošiljanja elektronskih sporočil




Slika 6.3: Memini dodatek za brskalnik Google Chrome.

iz spletne aplikacije ponuja tudi platformo za spremljanje analitike poslanih sporočil. Vidimo lahko, koliko sporočil je bilo uspešno dostavljenih, koliko uporabnikov je sporočilo odprlo ter kliknilo na povezave v sporočilih, itd.

### 6.2.2 Dodatek za brskalnik Google Chrome

Dodatek za brskalnik Google Chrome je kombinacija HTML in JavaScript kode. Ob kliku na ikono dodatka se uporabniku odpre obrazec (slika 6.3), ki vsebuje naslov in URL trenutne spletne strani ter besedilo, ki ga je uporabnik predhodno označil. Uporabnik se lahko tudi odloči, če želi za ta članek vklopiti opomnike.

Ob kliku na gumb za vložitev obrazca se sproži AJAX klic, ki podatke pošlje funkciji `save_from_chrome` v naši spletni aplikaciji. Če članek z enakim URLjem za tega uporabnika v podatkovni bazi že obstaja, funkcija le shrani izbrano besedilo k obstoječemu članku. Če članek še ne obstaja pa funkcija pred tem preko Readability APIja dobi besedilo in ostale ključne podatke članka ter jih shrani v bazo.

## 6.3 Testiranje

Python standardna knjižnica vsebuje modul `unittest`, ki je namenjen izvajanju integracijskih in unit testov. Django lahko med testiranjem simulira zahtevke (angl. *request*), v bazo vstavlja testne podatke in nadzoruje izhod aplikacije.

Tekom razvoja NSPja smo testirali predvsem kritične in komplicirane komponente, za katere je zelo verjetno, da bodo dolgo časa ostale del aplikacije. Testirali smo manj, kot bi pri klasičnem pristopu, saj smo se osredotočali na gradnjo pravega produkta, ne na pravilno gradnjo produkta. Šele, ko nam je znano, da gre produkt v pravo smer, se osredotočimo na najvišjo kvaliteto napisane kode. Kot je Mark Zuckerberg zapisal v pismu investitorjem, se je na začetku potrebno premikati hitro in pri tem tudi kaj polomiti (angl. *move fast and break things*). [36]

Pri testiranju smo se torej osredotočali na korake, ki so za uporabnike najbolj kritični. To so:

- Registracija in avtentikacija
- Shranjevanje različnih člankov
- Shranjevanje povzetkov
- Pregled povzetkov

## 6.4 Vzdrževanje

Za vzdrževanje infrastrukture nam ni potrebno skrbeti, saj za to poskrbijo vzdrževalci platforme Heroku. V primeru potrebe po bolj zmogljivi infrastrukturi lahko proti plačilu to nadgradimo z le nekaj kliki oz. ukazi v ukazni vrstici.

Za spremljanje nenavadnega dogajanja in odkrivanje možnih hroščev v aplikaciji uporabljamo Heroku dodatek Logentries [37], ki zapisuje vse dogajanje na strežniku in nam pošilja opozorila v primeru napak ali neodzivnosti.

Tako lahko analiziramo nepredvidene dogodke in odpravimo napake, ki so povzročile takšno obnašanje aplikacije.

Pozorni moramo biti na posodobitve Pocket in Readability APIja ter brskalnika Google Chrome. To so tehnologije, od katerih je odvisno delovanje naše aplikacije. V primeru večjih sprememb bi se lahko tudi v naši aplikaciji pojavile težave z delovanjem.

Nadgrajevanje aplikacije z novimi funkcionalnostmi poteka predvsem v skladu z obnašanjem in povratnimi informacijami uporabnikov. Z uporabniki moramo vedno ostati v tesnem stiku in analizirati njihov stil uporabe aplikacije, jo na podlagi tega nadgrajevati ter neprestano testirati novosti.

# Poglavje 7

# Sklepne ugotovitve

V sklopu diplomskega dela smo razvili spletno aplikacijo, ki se je iz razmeroma skromne in preproste ideje razvila v produkt z zelo smiselno funkcionalnostjo in interesom na trgu. S principi metodologije vitki startup nam je uspelo slediti potrebam na trgu, jih kombinirati z raziskavami o efektivnem učenju in razviti produkt, ki je v svojih zgodnjih fazah dosegel razultate nad pričakovanji.

## 7.1 Učne lekcije

**Zanko naredi-meri-spoznaj lahko pričnemo tudi z zadnjim korakom**
Internet je poln raziskav in preučevanja primerov, iz katerih se lahko veliko naučimo, predno postavimo hipoteze. Zanko torej lahko začnemo s korakom "spoznaj", v katerem raziščemo relevantne študije. To nam lahko prihrani veliko časa, a moramo biti pazljivi, da študije zares obravnavajo zelo podobno problematiko v sorodnih okoliščinah.

**Branje na mobilnih napravah je sedanjost, ne prihodnost**
Našo pristajalno stran je z mobilnih naprav obiskala približno tretjina obiskovalcov, globalni trendi pa so še precej višji, sploh na območju Azije. Tudi v ZDA potrošniki konzumirajo več medijskih vsebin preko mobilnih kot preko

namiznih naprav [38]. Hitra raziskava med znanci pozne generacije Y (rojeni po letu 1990) nam je razodela, da je razmerje uporabe med mobilnimi in namiznimi napravami pogosto kar 9:1. Glede na te podatke bi bilo smiselno testirati trg tudi za mobilne naprave.

**Stik s trgom nam poleg testiranja trga odpre tudi številne druge priložnosti**

Poleg potrjenih ali zavrženih hipotez ob stiku s trgom pridemo tudi v stik s posamezniki, ki nam vede ali nevede dajo nove ideje in odprejo nove poti. V našem primeru je bilo to na primer vabilo v podjetniški pospeševalnik ter pismo ameriškega študenta, ki je porodilo idejo za nov segment uporabnikov.

**Zelo enostavno je posvetiti preveč časa razvoju NSPja**

Najosnovnejši sprejemljivi produkt, ki smo ga razvili po testiranju Evernote prototipa, je imel kar 14 funkcionalnosti. Ves čas smo se namreč osredotočali na prijetno uporabniško izkušnjo. Ena izmed manj nujnih funcionalnosti je na primer arhiv, ki smo ga implementirali zato, da uporabniki lahko odstranjujejo elemente iz seznama za branje in se s tem izognejo neurejenemu in predolgemu seznamu. A ob prvotnem testiranju na trgu to najbrž ni igralo bistvene vloge.

**Uporaba striktnih rokov in ciljev pomaga, da v NSP ne vložimo preveč časa**

Čeprav smo se zavedali, da mora NSP vsebovati le bistvene funkcionalnosti, smo se tudi sami ujeli v past in skušali implementirati lastnosti, ki niso nujno potrebne. Opazili smo, da se tega najlažje rešimo s striktnimi roki in cilji. Zastaviti si moramo plan, kdaj bomo lansirali NSP in katere nujno potrebne funkcionalnosti bo imel.

**Včasih se je potrebno oddaljiti od razvoja in produkt pogledati iz drugačne perspektive**

Ko je bil NSP že skoraj dokončan, smo zaradi nepričakovanih okoliščin za 3 mesece ustavili razvoj. Ko smo se vrnili k razvoju, smo na produkt gledali mnogo bolj objektivno in lažje smo se vživeli v vlogo naših uporabnikov. To nam je porodilo ideje za mnoge spremembe, ki so naredile uporabo produkta bistveno bolj enostavno in intuitivno.

## 7.2 Nadaljnji razvoj

Cilj naloge je bil razviti aplikacijo do faze NSP, a glede na to, da ima produkt nekaj aktivnih uporabnikov ter še precej registriranih uporabnikov, ki zaradi omejitve niso dobili dostopa do uporabe, bi bilo smiselno produkt dodelati vsaj do te mere, da bo ponujal vse smiselne funkcionalnosti, ne le najnujnejših, in pri tem čim manjkrat naletel na napako.

Med manjše a nujne izboljšave spada obdelava in reševanje vseh izjem in napak pri izvajanju programske kode, izpopolnitev registracije novih uporabnikov (ta trenutno poteka brez preverjanja avtentičnosti elektronske pošte), funkcionalnost za pozabljeno geslo, upoštevanje različnih časovnih pasov uporabnikov ter možnost shranjevanja slik v izpiskih.

Poleg povezave z aplikacijo Pocket bi, v primeru interesa s strani uporabnikov, lahko dodali tudi povezavo s sorodnimi aplikacijami, kot sta Readability in Instapaper. Pri tem bi bilo morda smiselno implementirati selektivno sinhronizacijo na podlagi značk – to bi uporabnikom omogočilo, da ob shranjevanju vsakemu članku dodajo vnaprej določeno značko, ki določa, če ga želijo sinhronizirati z našo aplikacijo.

Namesto fiksnih period za ponavljanje s presledki, bi bila bolj učinkovita implementacija dinamičnih personaliziranih period. Uporabnik bi med ponavljanjem za vsak izpisek izbral, kako dobro si ga je zapomnil od zadnje ponovitve in glede na odgovor bi aplikacija določila datum naslednje ponovitve prej ali kasneje v prihodnosti. Lahko bi datum ponovitve izbral tudi uporabnik sam. Za še mnogo boljši učinek pa bi morali implementirati aktivno

učenje, kjer bi uporabniki namesto pasivnega prebiranja zapiskov morali le-te aktivno priklicati iz spomina.

Med testiranjem smo ugotovili, da navkljub zelo enostavni uporabniški izkušnji uporaba aplikacije še vedno zahteva precej samodiscipline, predvsem, ko je potrebno prebrati vsebino opomnikov. Ker je to bistvo celotne aplikacije, bomo uporabnike najbrž izgubljali, če jim ne bomo tega procesa poenostavili. Izboljšava bi bila najbrž možna z implementacijo igrifikacije (angl. *gamification*) ter oblikovanja uporabniške izkušnje po modelu ATARI (angl. *A Trigger, Action, Reward, Investment* – sprožilec, dejanje, nagrada, investicija) za spodbujanje navad [39].

Trenutno smo dokaj omejeni z uporabo dodatka za Chrome, saj je za uporabnike vseh ostalih brskalnikov aplikacija le delno uporabna. Google Chrome sicer drži skoraj 62 % tržni delež [40], kar je bil glavni povod, da smo se odločili za ta brskalnik, a vseeno nas to ne sme omejevati. Za namizne brskalnike bi bila najbolj preprosta začasna rešitev izdelava aktivnega zaznamka (angl. *bookmarklet*), ki bi deloval v večini brskalnikov. Kasneje bi se seveda lahko lotili izdelave dodatkov za posamezne brskalnike, kar bi bilo uporabnikom mnogo bolj prijazno, a bi tudi zahtevalo več razvoja in vzdrževanja.

Nikakor pa ne smemo pozabiti tudi na mobilne naprave. Inoviranje in nadaljnje testiranje trga v tej smeri bi bilo še posebej zanimivo, saj je trenutno izdelava zapiskov med branjem na mobilnih napravah relativno nepraktična.

# Literatura


[1] (2015) Mapping Tech Startups. Allmand Law. Dostopno na: https://allmandlaw.com/articles/mapping-tech-startups.

[2] (2015) Lean manufacturing. Wikipedia. Dostopno na: http://en.wikipedia.org/wiki/Lean_manufacturing.

[3] (2015) United States CTO Todd Park - Lean Startup Conference 2012 HD. Youtube. Dostopno na: https://www.youtube.com/watch?v=8WLM61QwbvU.

[4] A. Osterwalder in Y. Pigneur, *Business Model Generation*, New Jersey: John Wiley & Sons, 2010.

[5] J. P. Womack in D. T. Jones, *Lean Thinking: Banish Waste and Create Wealth in Your Corporation*, New York: Productivity Press, 2003.

[6] R. Fitzpatrick, *The Mom Test: How to talk to customers & learn if your business is a good idea when everyone is lying to you*, CreateSpace Independent Publishing Platform, 2013.

[7] (2015) Apple Music Event 2001-The First Ever iPod Introduction. Youtube. Dostopno na: https://www.youtube.com/watch?v=kN0SVBCJqLs.

[8] (2015) Attention Span Statistics. Statistic Brain. Dostopno na: http://www.statisticbrain.com/attention-span-statistics/.

[9] (2015) Landing Page Design and How to Use it to Sell Your Indie Game. Indie Game Girl. Dostopno na: http://www.indiegamegirl.com/landing-page-design-and-how-to-use-it-to-sell-your-indie-game/.

[10] J. Lecinski, *Winning the Zero Moment of Truth – ZMOT*, Vook, 2011.

[11] P. H. Diamandis in S. Kotler, *Bold: How to Go Big, Create Wealth and Impact the World*, New York: Simon & Schuster, 2015.

[12] (2015) "Fail fast" advises LinkedIn founder and tech investor Reid Hoffman. BBC News. Dostopno na: http://www.bbc.co.uk/news/business-12151752.

[13] A. Maurya, *Delaj Vitko: Od načrta A do načrta, ki deluje*, Ljubljana: Založba Pasadena, 2014.

[14] (2015) Startup Metrics for Pirates (Lean Startup Circle, Jan. 2010). Master of 500 Hats. Dostopno na: http://500hats.typepad.com/500blogs/2010/01/startup-metrics-for-pirates-lean-startup-circle-jan-2010.html.

[15] (2015) Google Analytics – Bounce Rate: The Simply Powerful Metric. Youtube. Dostopno na: https://www.youtube.com/watch?v=ppgfjo6IIf4.

[16] S. Godin, *Purple Cow: Transform Your Business by Being Remarkable*, Portfolio, 2003.

[17] S. G. Blank, *The Four Steps to the Epiphany*, K&S Ranch, 2013.

[18] (2015) How to bring a product to market / A very rare interview with Sean Ellis. Venture Hacks. Dostopno na: http://venturehacks.com/articles/sean-ellis-interview.

[19] (2015) Combining agile development with customer development. Startup Lessons Learned. Dostopno na: http://www.startuplessonslearned.com/2009/03/combining-agile-development-with.html.

[20] (2015) Agile Software Development Benefits. VersionOne. Dostopno na: http://www.versionone.com/agile-101/agile-software-development-benefits/.

[21] E. Ries, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, New York: Crown Business, 2011.

[22] D. J. Anderson, *Kanban: Successful Evolutionary Change for Your Technology Business*, Blue Hole Press, 2010.

[23] R. E. Bohn in J. E. Short, *How Much Information? 2009 Report on American Consumers*, Global Information Industry Center, University of California, San Diego, 2009.

[24] (2015) Introduction to Neuroeconomics: how the brain makes decisions. Coursera. Dostopno na: https://www.coursera.org/course/neuroec.

[25] J. Michael, Where's the evidence that active learning works? *Advances in Physiology Education 30(4):159-167*, American Physiological Society, 2006.

[26] (2015) Take your time: Neurobiology sheds light on the superiority of spaced vs. massed learning. Medical Xpress. Dostopno na: http://medicalxpress.com/news/2012-03-neurobiology-superiority-spaced-massed.html.

[27] (2015) Heroku Postgres. Heroku dev center. Dostopno na: https://devcenter.heroku.com/articles/heroku-postgresql.

[28] (2015) How Heroku Works. Heroku dev center. Dostopno na: https://devcenter.heroku.com/articles/how-heroku-works.

[29] (2015) How a Two-Day Side Project Turned Into a Productivity App with 11 Million Users. Zapier Blog. Dostopno na: https://zapier.com/blog/pocket-nate-weiner-interview/.

[30] (2015) What Is the Average Reading Speed and the Best Rate of Reading? Healthguidance.org. Dostopno na: http://www.healthguidance.org/entry/13263/1/What-Is-the-Average-Reading-Speed-and-the-Best-Rate-of-Reading.html.

[31] (2015) Curve of Forgetting. University of Waterloo. Dostopno na: https://uwaterloo.ca/counselling-services/curve-forgetting.

[32] (2015) Memini. Wiktionary. Dostopno na: http://en.wiktionary.org/wiki/memini.

[33] (2015) Heroku Scheduler. Heroku add-ons. Dostopno na: https://addons.heroku.com/scheduler.

[34] (2015) Heroku Scheduler. Heroku dev center. Dostopno na: https://devcenter.heroku.com/articles/scheduler.

[35] (2015) Mandrill by MailChimp. Heroku dev center. Dostopno na: https://devcenter.heroku.com/articles/mandrill.

[36] (2015) Mark Zuckerberg's Letter to Investors: "The Hacker Way". Wired. Dostopno na: http://www.wired.com/2012/02/zuck-letter/.

[37] (2015) Logentries. Heroku dev center. Dostopno na: https://devcenter.heroku.com/articles/logentries.

[38] (2015) Mobile Marketing Statistics 2015. Smart Insights. Dostopno na: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/.

[39] N. Eyal, *Hooked: How to Build Habit-Forming Products*, Portfolio, 2013.

[40] (2015) Browser Statistics. w3schools.com. Dostopno na: http://www.w3schools.com/browsers/browsers_stats.asp.